SMART TRAFFIC LIGHT CONTROL SYSTEM USING IMAGE PROCESSING

PROJECT REPORT

Submitted by

SAMITHA R.	**(19BCS005)**
NIKESH R.	**(19BCS093)**
TAMILARASAN D.S.	**(19BCS099)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DR. MAHALINGAM COLLEGE OF

ENGINEERING AND TECHNOLOGY

POLLACHI – 642 003

(An Autonomous Institution Affiliated to Anna University, Chennai)

MAY 2023

Dr. MAHALINGAM COLLEGE OF ENGINEERING AND TECHNOLOGY, POLLACHI -642003

(An Autonomous Institution Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this Project Report, "SMART TRAFFIC LIGHT MONITORING SYSTEM USING IMAGE PROCESSING"
is the bonafide work of

SAMITHA R.	(19BCS005)
NIKESH R.	(19BCS093)
TAMILARASAN D.S.	(19BCS099)

who carried out the project work under my supervision.

Dr.M.Pandi SUPERVISOR Associate Professor Computer Science and Engineering Dr. Mahalingam College of Engineering and Technology, NPT-MCET Campus Pollachi – 642 003 India	Dr.G.Anupriya HEAD OF THE DEPARTMENT Professor Computer Science and Engineering Dr. Mahalingam College of Engineering and Technology, NPT-MCET Campus Pollachi – 642 003 India

Submitted for the Autonomous End Semester Examination Project Viva-voce held on

____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

SMART TRAFFIC LIGHT MONITORING SYSTEM USING IMAGE PROCESSING

ABSTRACT

The movement of cars through intersections where several roads converge is frequently tracked and managed by traffic signal control systems. But considering the different characteristics involved, coordinating numerous systems of traffic lights at nearby junctions is a difficult task. The main focus of traditional approaches is giving all lanes a comparable amount of time. Additionally, the current traffic system does not take into account the reciprocal interaction among nearby traffic light networks, the variation in car flows over time, collisions, the approaching of emergency vehicles, or crosswalks for pedestrians. Traffic congestion and delays result from this. Road traffic management has been utilised as a means of ensuring the security of both vehicles and pedestrians. Therefore, in order to minimise the amount of individual fatalities, it is crucial to begin developing an algorithm as rapidly as practicable. To identify and categorise Low, Medium, and Heavy Traffic in the movement of highway traffic in actual time, however, currently does not exist modern technology that is as reliable as it needs to be. In order to solve this issue, the study recommends using road images for refining a machine learning model for recognising traffic and categorization. From collections of actual road visual data, the Convolutional Neural Network (CNN) model has been instructed to recognise and classify traffic. The CNN approach, that is based on the design of Xception and employs the ADAM optimizer, is recommended for improved categorization. Each lane's traffic volume is categorised as low, medium, or high according to the CNN algorithm. The microcontroller chooses these levels depending on information from the CNN model. The Microcontroller assigns durations for the lanes and modifies the red, yellow, and green signals based on the volume of traffic. The possibility for employment in a fast and precise identification of traffic on the roadways is shown by the suggested method's efficiency, which outperforms the current version of research.

ACKNOWLEDGEMENT

First and foremost, we wish to express our deep unfathomable feeling, gratitude to our institution and our department for providing us a chance to fulfill our long-cherished dreams of becoming Computer Science Engineers.

We express our sincere thanks to our honorable Secretary **Dr. C. Ramaswamy** for providing us with required amenities.

We wish to express our hearty thanks to **Dr.P. Govindasamy**, Principal of our college, for his constant motivation and continual encouragement regarding our project work.

We are grateful to **Dr. G. Anupriya**, Head of the Department, Computer Science and Engineering, for her direction delivered at all times required. We also thank her for her tireless and meticulous efforts in bringing out this project to its logical conclusion.

Our hearty thanks to our guide **Dr.M.Pandi,** Associate Professor for his constant support and guidance offered to us during the course of our project by being one among us and all the noble hearts that gave us immense encouragement towards the completion of our project.

We also thank our review panel members for their continuous support and guidance.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CO2	Carbon dioxide
DL	Deep Learning
Deep SORT	Deep Simple Online and Real-Time Tracking
GCN	Graph Convolutional Network
ITS	Intelligent Transport System
IOT	Internet of Things
MLP-NN	Multilayer Perceptron Neural Network
NN	Neural Network
RAM	Random Access Memory
RNN	Recurrent Neural Network
YOLO	You Only Look Once

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The number of automobiles on the road has suddenly increased throughout the years. Everyone now has to deal with the rising issue of traffic congestion on a daily basis. The number of automobiles on the road has suddenly increased throughout the years. Everyone now must deal with the growing issue of traffic congestion daily. The effectiveness of traffic-by-traffic police manual control has not been established. Moreover, the predetermined set the signal's timer under all conditions (low and high traffic density) has not provided a solution to this issue. Machine learning (ML), a branch of artificial intelligence, is currently one of the most prominent and well-known expanding fields (AI). Recently, machine learning has become a vital and exciting research area in transportation engineering, notably in the area of traffic forecasting. Since traffic congestion affects people from all social strata equally, it is necessary to estimate traffic on a small scale so that people can live their lives without stress or annoyance. To ensure the country's economic advancement, road users' convenience comes first and foremost. Only an ongoing traffic flow makes this feasible. Traffic forecasting is required to address this issue since it allows us to roughly anticipate or forecast future traffic.

The government additionally makes investments in intelligent transportation systems to address these issues (ITS). The primary objectives of this study are to categorise various machine learning approaches and speculatively develop Python 3 models. Predicting traffic flow's objective is to provide users with a quick estimate of traffic. Hence, traffic predictions could benefit from this research. The Internet of Things (IoT) is presented as a solution to the issues. We employ machine learning to compute and process the data, and an IOT-based system and a Processing application are used to display the results. The goal of this project is to create an image processing based IOT traffic monitoring and control system. By managing the traffic lights, the system will clear the way for vehicles. The system utilises a cutting-edge concept for an IOT-based traffic monitoring and control system. With the aid of a central unit, we will use this technology to regulate each city path's traffic light. To control the system, an ESP32 microcontroller will be used in the central unit.

1.1 PROBLEM DESCRIPTION

Real-time traffic detection addresses various issues that are interconnected in a hierarchy, predicts traffic flows at the proper resolution levels (individual vehicles and platoons) to enable proactive control, allows for various optimisation modules for solving the hierarchical sub problems, and uses a data structure and computer/communication approaches that allow for quick solution of the sub problems so that each decision can be made with enough time.

1.2 OBJECTIVE OF THE PROJECT

The project's main objective is to identify traffic on the highways from the collected real time images using a CNN-based model. To achieve this, databases of road picture data are employed to train the image classification model. We post real-time road images to categories the amount of traffic as Low, Medium, Heavy, or No Traffic. The Microcontroller assigns a lane a light and modifies the red, yellow, and green indications according to the volume of traffic. The lane lights alter, according on the classified image.

1.3 SCOPE OF THE PROJECT:

These days, the classification model's performance is likewise biased, and it is claimed to have used sustained vowels to detect traffic with accuracy of 98% and 97.5%, respectively, and noted that the proposed validation approach is speaker dependent. Although the training and test sets of the road datasets are in contrast with one another, this validation method may produce an overly optimistic outcome. According to the research, if the training epochs are reduced, accuracy can even drop to 60%.

The goal of traffic detection systems is to eliminate the need for manual labor to automate or computerize routine vision-based tasks. Computer vision systems have also been used in several public spaces, including highways, airports, and shopping malls. The duty of monitoring and analyzing traffic scenes, with a focus on highways and intersections, is one such application of vision systems. Such a system is necessary for efficient real-time traffic management systems that can immediately identify changes in traffic characteristics and enable regulators and authorities to react to traffic problems.

CHAPTER 2

LITERATURE SURVEY

2.1 DEVELOPMENT OF A SMART TRAFFIC LIGHT CONTROL SYSTEM WITH REAL-TIME MONITORING

Traffic signal management programmes have routinely utilised to observe and manage the flow of cars since they first appeared. Nonetheless, metropolitan centres are populating more and more due to the rise in the number of private and public transportation options (bus, automobile, motorcycle, and truck). Such a phenomena increases environmental and noise pollution as well as transportation congestion. Large cities are using technological solutions to address these issues, bringing the idea of "smart cities" to life. Many hardware and software solutions have been researched and put into practise all around the world while keeping an eye on traffic control systems themselves. By creating a centralised Using a specialised wireless communication network and a traffic light management system, the author of this essay hopes to improve traffic lights. The most frequent kinds of urban junctions were examined to demonstrate the system's efficacy. To provide a complete set of controls for anomalous events, such as stopping roads due to accidents or public gatherings, direct control procedures for network traffic lights were built. Eventually, safety procedures were developed to inform central management of the traffic light system bulbs' operational condition. It was feasible to make up an operational using a logic analyser connected to the outputs for each focus group, create an operational phases timing diagram for each traffic light. Consequently, parallels between theoretical and real timing diagrams were used to validate the system.

2.2 SMART CONTROL OF TRAFFIC LIGHT USING ARTIFICIAL INTELLIGENCE

Traffic congestion is one of the most urgent issues due to a growth in both the population and the usage of cars in cities. Traffic jams not only prolong commute times and raise stress levels for drivers, but they also considerably increase air pollution and fuel consumption. While appearing to be everywhere, it has the greatest influence on megacities. Due to the ever-increasing nature of road traffic, it is also crucial to estimate the road traffic density in real-time for enhanced signal control and effective traffic management. One of the important elements influencing traffic flow is the traffic controller. As a result, it becomes necessary to optimize traffic management to better meet this rising demand. Our suggested method intends to use real-time photos from the cameras at traffic intersections for image processing and AI-based traffic density estimation. It also focuses on the algorithm for changing traffic signals depending on vehicle density in order to relieve congestion, speed up travel for commuters, and lessen pollution.

2.3 A REAL-TIME DENSITY-BASED TRAFFIC SIGNAL CONTROL SYSTEM

Accidents and traffic congestion are becoming two of the biggest problems in Sri Lanka. Many social, economic, and environmental challenges result from these issues. One of the causes is a lack of an efficient traffic light control system. This study suggested a method for creating a real-time traffic signal based on density management system. The two main components of this study are an ANN model for real-time data prediction and a real-time data collection paradigm for image processing. In order to reduce the dimensionality of the features and find the best features from the acquired data, principal component analysis (PCA) is used to train neural network models. Lanes are monitored and photographed using cameras. The number of vehicles in each lane and the duration of the queue are identified and counted using image processing. The data from each lane is sent to the ANN unit. Based on the amount of automobiles, the trained model will decide what lanes and time restrictions are necessary to enable the green phase. The NN model's accuracy during training was 0.9274. The traffic lights at the intersections will have changed independently and dynamically in line with the conditions of real-time traffic when using this traffic light control system instead of the present fixed time traffic light control system or traditional computer approaches. This method reduces the typical waiting time while increasing the effectiveness of the traffic flow. New adaptive traffic management also lessens pollution brought on by CO2 emissions as well as social and economic issues.

2.4 ADAPTIVE TRAFFIC LIGHT CONTROLLER SIMULATION FOR TRAFFIC

The control of traffic lights becomes more significant as the number of cars rises in densely populated places. Using adaptive traffic lights, traffic signals may change the flow of traffic in accordance with the number of cars in each segment. This research focused on modelling based on the Unity3D platform employing fuzzy logic control and advocated the usage of an agent-based modelling system to regulate traffic light. Three different environments—busy, moderate, and smooth were used for the test. The average waiting time, system performance is measured by the average number of vehicle stops and the average vehicle speed. metrics. According to simulation data, adaptive traffic lights can enhance vehicle speed while decreasing waiting times and the number of cars that halt at intersections.

2.5 VISION-BASED ADAPTIVE TRAFFIC LIGHT CONTROLLER FOR SINGLE INTERSECTION

In this study, a vision-based adaptive traffic signal controller is proposed. The planned controller was successfully installed and tested as a whole system in a challenging Colombo roundabout at a time when traffic was at its worst. There were two main parts to its implementation. The first was a system for vision-based traffic monitoring. This section describes a system that employed cameras to keep an eye on the lanes at a junction. The video feeds were then processed to build a traffic index based on variables including vehicle type, traffic density, and pixel-wise vehicle velocity. The second component of the project was the controlling of traffic signal lights. In this section, we used a mathematical modelling technique to estimate a better time modification for the present system. For simple implementation in the actual world, this system was operated using the current system with few modifications. To adjust traffic signal phase timing in accordance with the volume of existing traffic, the produced prototype was connected into the existing system.

2.6 TRAFFIC FLOW PREDICTION FOR SMART TRAFFIC LIGHTS USING MACHINE LEARNING ALGORITHMS

Nowadays, many cities have traffic congestion during certain peak hours, which raises locals' exposure to pollution, noise, and stress. Neural networks (NN) and machine-learning (ML) approaches are more frequently used to address real-world problems than analytical and statistical approaches due to their ability to handle dynamic behaviour over time and with a variety of parameters in large amounts of data. This study proposes machine learning (ML) and deep learning (DL) algorithms for predicting traffic flow at an intersection, laying the groundwork for adaptive traffic control, either by applying an algorithm that modifies the timing of the traffic lights in accordance with the predicted flow or by remotely controlling traffic lights. Hence, forecasting traffic flow is the only goal of this effort. Two open datasets are used to train, validate, and test the suggested ML and DL models. The first one comprises a sample of all the vehicles collected over a period of 56 days at six intersections using different sensors. The Recurrent Neural Networks (RNNs), which had high metrics outcomes but took more training time, came in second, followed by Gradient Boosting, Recurrent Neural Networks, Linear Regression, and Stochastic Gradient, and lastly Random Forest, Linear Regression, and Stochastic Gradient. Better results (R-Squared and EV score of 0.93) were obtained with the Multilayer Perceptron Neural Network (MLP-NN), which also needed less training time.

2.7 AN EDGE TRAFFIC FLOW DETECTION SCHEME BASED ON DEEP LEARNING IN AN INTELLIGENT TRANSPORTATION SYSTEM

Intelligent transport systems have a considerable influence on the management of public transit, security, and other issues (ITS). Sensing of traffic movement is a crucial part of the ITS. Based on the real-time data collection of information on urban road traffic flow, an ITS delivers intelligent recommendations for reducing environmental pollution and traffic congestion. Traffic flow detection is frequently done using cloud computing. The edge of the network will transmit all the captured video to the cloud computing facility. Nevertheless, due to the increasing traffic monitoring, conventional transport systems based on cloud computing have encountered major difficulties in terms of storage, connection, and processing .In this research, a deep learning-based edge node traffic flow detection approach is proposed as a remedy for this issue. The YOLOv3 (You Only Look Once) model, which has been extensively trained using traffic data, is the basis of the second approach we provide, which is for the detection of moving objects. We reduced the model to ensure its performance on modern hardware. The DeepSORT (Deep Simple Online and Real-time Tracking) technique is then optimized when the feature extractor is retrained for multi object vehicle tracking. Next, we describe a real-time vehicle tracking counter for autos that combines vehicle recognition and vehicle tracking algorithms to achieve the identification of traffic flow. In the end, we deploy and migrate the multiple objects tracking network and vehicle detection network onto the Jetson TX2 platform for edge devices, and we evaluate the precision and efficiency of our framework. The test results show that our model, with an average processing speed of 37.9 FPS (frames per second) and a processing speed of 92.0%, can accurately and successfully detect traffic flow on edge devices.

2.8 INTELLIGENT DRIVER DROWSINESS DETECTION FOR TRAFFIC SAFETY BASED ON MULTI CNN DEEP MODEL AND FACIAL SUBSAMPLING

According to statistics, sleepy, tired, or distracted driving is a major contributor to accidents on the world's highways. Several research on the issue of automated drowsiness detection propose to collect physiological data from the driver, such as the ECG, EEG, heart variability rate, blood pressure, etc., making those approaches subpar. Recent ones propose computer vision-based remedies, their performance is restricted since they either employ manually created features with traditional methods like Naive Bayes and SVM or too complex deep learning models that still perform poorly. As a result, In this paper, we provide an ensemble deep learning architecture that evaluates the driver's fitness using a decision structure and integrated attributes from mouth and eye subsamples. The proposed ensemble model consists of only two InceptionV3 modules, which help to constrain the parameter space of the network. These two modules independently and unilaterally perform feature extraction for the mouth and eyes subsamples from the face images. The output of each output is passed to the ensemble boundary using the weighted average technique, whose weights are modified by the ensemble process. The output of this system determines whether the driver is tired or not. The benchmark NTHU-DDD video dataset is used to effectively train and assess the suggested model. With train and validation accuracy of 99.65% and 98.5%, respectively, the model achieved accuracy on the evaluation dataset of 97.1%, which is substantially higher than that of previously recommended models.

2.9 A PERFORMANCE MODELING AND ANALYSIS OF A NOVEL VEHICULAR TRAFFIC FLOW PREDICTION SYSTEM USING A HYBRID MACHINE LEARNING-BASED MODEL

Recently, a lot of focus has been placed on the Intelligent Transportation System (ITS), which incorporates traffic prediction on the road. It's always a hot topic to figure out how to develop an accurate, efficient, and trustworthy system for anticipating automobile traffic. The employment of machine learning-based (ML) techniques, particularly deep learning-based (DL) approaches, increases the prediction model's accuracy. Nevertheless, we also noted that there are still a great deal of problems with the use of ML-based vehicular traffic forecast models in the actual world. First off, the training period for DL models is much longer than it is for parametric models like ARIMA and SARIMA. Not to mention, it's essential that we put the prediction system to use while simultaneously figuring out how to leverage the cutting-edge technology employed in ITS to improve the prediction system itself. In this article, we focus on improving the prediction model's characteristics so that it can be applied in real-world scenarios. A unique hybrid deep learning model is presented using the Graph Convolutional Network (GCN) and the deep aggregation structure (i.e., the sequence-to-sequence structure) of Gated Recurrent Unit (GRU). To solve the real-world prediction issue, often known as the online prediction task, we present a unique online prediction approach based on refinement learning. To improve the model's accuracy and efficiency when employed with ITS, we apply a powerful parallel training approach and the vehicular cloud structure.

2.10 A SURVEY OF TRAFFIC PREDICTION: FROM SPATIO-TEMPORAL DATA TO INTELLIGENT TRANSPORTATION

Our trip is more convenient and effective thanks to intelligent transportation (such as an intelligent traffic light). It makes sense to collect spatio-temporal data and utilize it to promote the goal of intelligent mobility, where traffic prediction is vital, as mobile Internet and location technologies progress. We present a thorough overview of traffic prediction in this study, including everything from the spatiotemporal from the data layer to the application layer for intelligent transportation. Starting from the bottom and working our way up, we divided the entire study area into four sections: spatiotemporal data, preprocessing, traffic prediction, and traffic application. We then examine the previous research on the four sections. Second, based on variations in space and time, we divide traffic data into five types. We focus on four crucial data preparation techniques in the second section: map-matching, data cleaning, data storage, and data compression. Finally, we concentrate on the categorization, creation, and estimation/forecasting of three different traffic prediction issues. We list the issues and talk about how current approaches deal with them. Fifth, we provide a list of five common traffic applications. Finally, we offer recent research prospects and difficulties. We think that the survey can aid practitioners in comprehending the methodologies and issues associated with traffic prediction, which will further motivate them to develop their own intelligent transportation solutions.

CHAPTER 3

EXISITNG SYSTEM

Automated monitoring and surveillance systems aim to eliminate the requirement for human labor for straightforward vision-based activities that may be completed by a computer or an automated system. Computer vision systems have also been used in a number of public spaces, including highways, airports, and shopping malls. The duty of monitoring and evaluating traffic scenes, with a focus on highways and intersections, is one such use of vision systems. Such a system is necessary for efficient real-time traffic management systems that are able to immediately identify changes in traffic characteristics and enable regulators and authorities to react to traffic problems.

3.1 DISADVANTAGES OF EXISTING SYSTEM

- The Existing System requires a Large Manpower to monitor traffic.
- The Existing System has a huge Traffic Congestion.
- A huge amount of time is wasted in Monitoring the system.

CHAPTER 4

PROPOSED SYSTEM

This tactic exhibits a substantial CNN was used to categorize photos of moving automobiles into four groups: Low, Medium, High, and No Traffic. They demonstrate that cutting-edge outcomes may be obtained when deep CNNs are used to the classification of traffic types. CNN includes several steps (hidden layers) for feature extraction that may autonomously learn representations from the input making it the best and most effective image processing method available right now. Research on CNNs has been motivated by the development of some exciting deep CNN designs in recent years. Convolutional Neural Network (CNN) models can map input images to their labelled classes and demonstrate high test dataset generalization. The suggested CNN technique is effective for traffic detection and can carry out the task on four target traffic classes with a high degree of accuracy, according to experimental results on real-world datasets. IOT will update the state of the traffic lights.

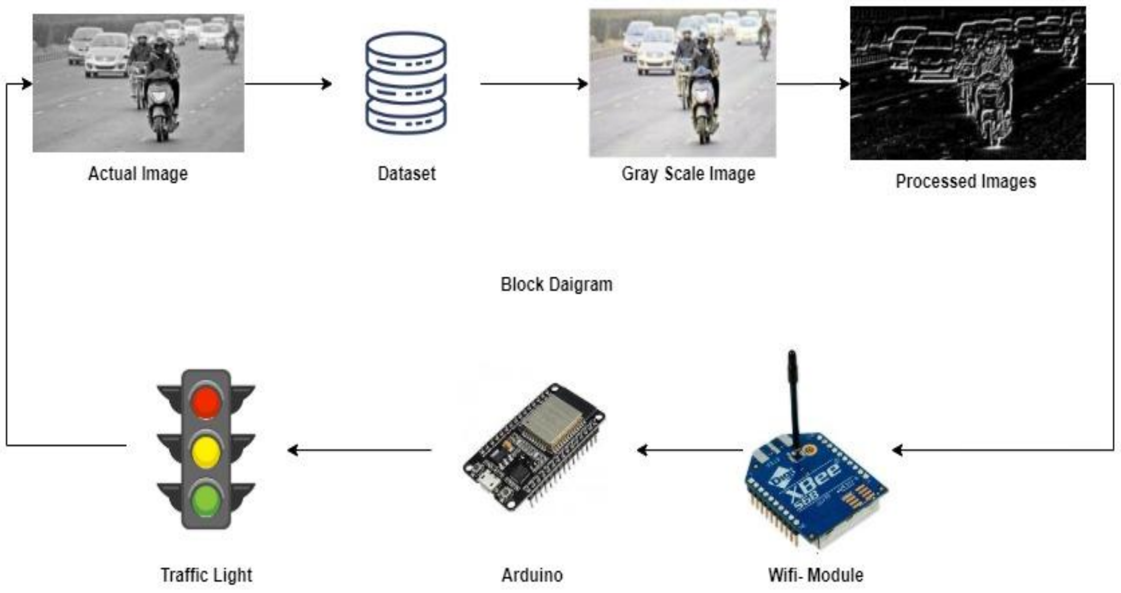


Figure 4.1: Process flow of the proposed system

ADVANTAGES:

- The Proposed System makes use of images for Easily Detection
- The Proposed System makes use of Low Manpower.
- The System provides a High Accuracy level.

4.1 LIST OF MODULES

4.1.1 IMAGE ACQUISITION:

The suggested suggestions have been implemented using the dataset's Real Time Road Pictures. This method, which incorporates several processes including image capture, edge recognition, and traffic detection classification, is used to design for the exact extraction of traffic.

4.1.2 IMAGE PRE-PROCESSING:

In this module, we execute a few fundamental actions on images in order to provide the best image for processing. To create a suitable and clean image, we will carry out a number of operations in this module, including grayscale conversion, filtering, sharpening, smoothing, edge, and image segmentation. By lowering noise, preprocessing improves the quality of the photos. Grayscale images, which are a subset of monochromatic or black-and-white photographs, only contain the color grey. Grey scale photographs allow for the evaluation of pixel brightness. The filtering procedure is used to improve the image's edges, smoothness, and sharpness. With a sharpening filter, the photographs' sharpness and the clarity of hazy components are both improved. Smoothing filters are noise-cancelling filters. a variety of algorithms. A method for detecting and distinguishing sharpness in a photo is edging.

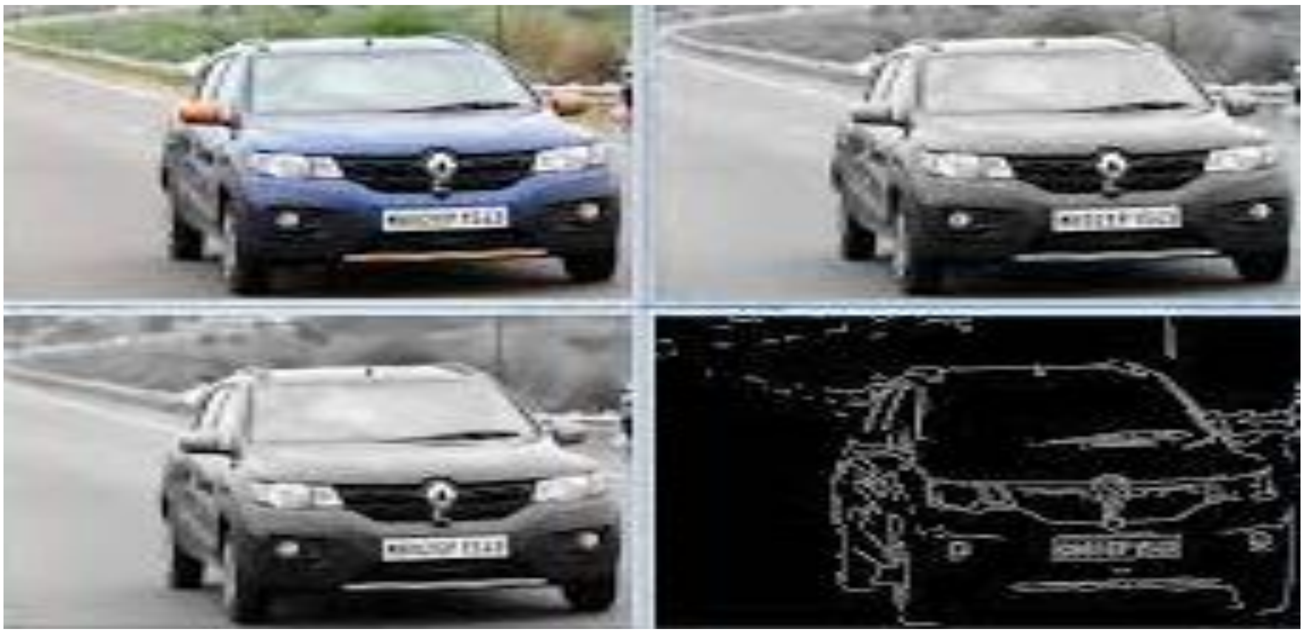


Figure:4.2 Image Preprocessing

4.1.3 IMAGE SEGMENTATION:

Recent developments in applications such as imaging, video surveillance, and many other fields, image segmentation is a crucial step in the science of computer vision. At the image segmentation stage of processing, the entire picture is split into binary images using the threshold approach. Digital images may be divided up into many different parts or items. Segmenting involves assembling groups of pixels with related characteristics. It is used to identify borders and objects in pictures. In essence, the segmentation procedure is used to remove crucial elements from the image for additional analysis.

4.1.4 FEATURE EXTRACTION:

We are doing further operations on the segmented picture in this module. We'll use a feature extraction approach in this module to get all the information we want about the traffic image. In computer vision and machine learning, feature extraction and reduction have proven crucial for categorization traffic identification. The main challenge in feature extraction is deciding which traits are most relevant or trustworthy for categorization and which result in meaningful outcomes. When dimensionality is decreased, feature extraction is used.

4.1.5 TRAFFIC CLASSIFICATION:

Using deep learning techniques, we develop classification strategies in this session to assess the state of the road traffic. At the final step of the recommended technique, road traffic is divided into several categories, including Low Traffic, Medium Traffic, Heavy Traffic, and No Traffic. Following feature selection and extraction, CNN classification is next performed to the resulting feature vector. The categorising process uses the training and testing phases of the CNN framework.

4.2 FLOW CHART OF THE PROPOSED SYSTEM:

4.2.1 TRAFFIC DETECTION SYSTEM:

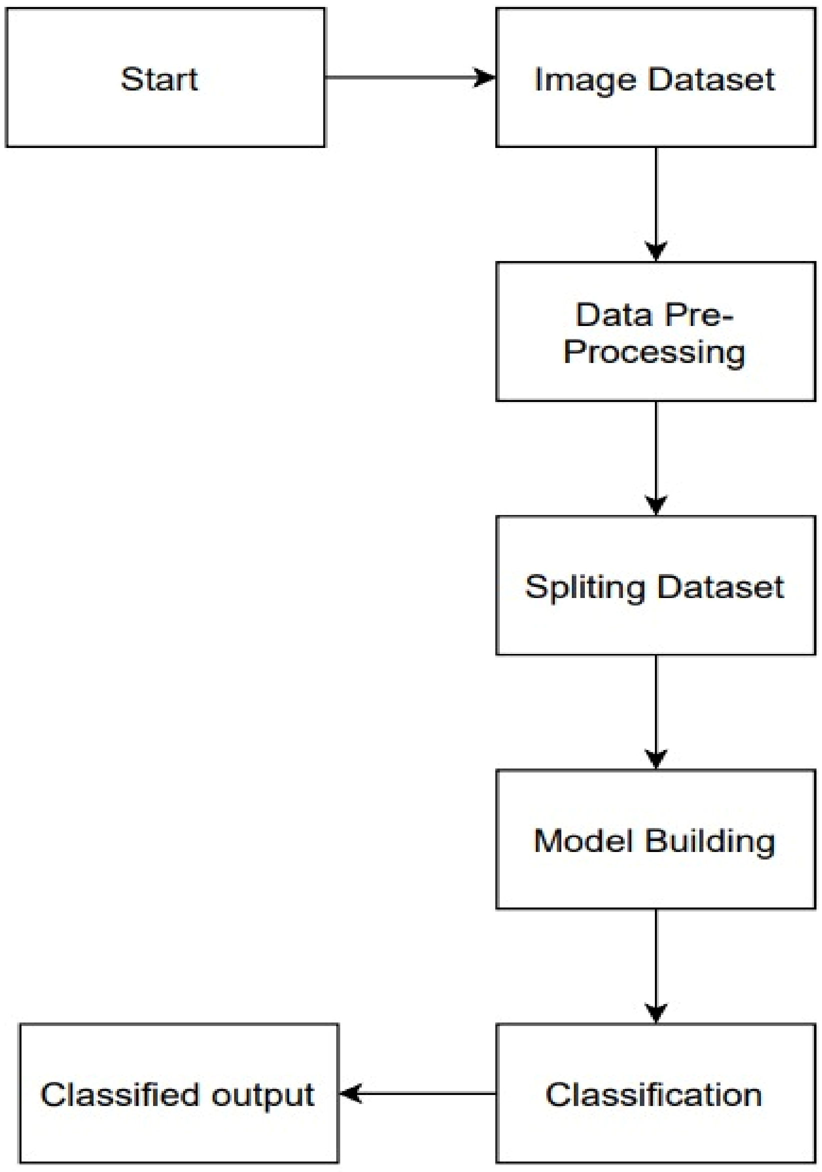


Figure:4.3 Traffic Detection System

4.2.2 BLOCK DIAGRAM:

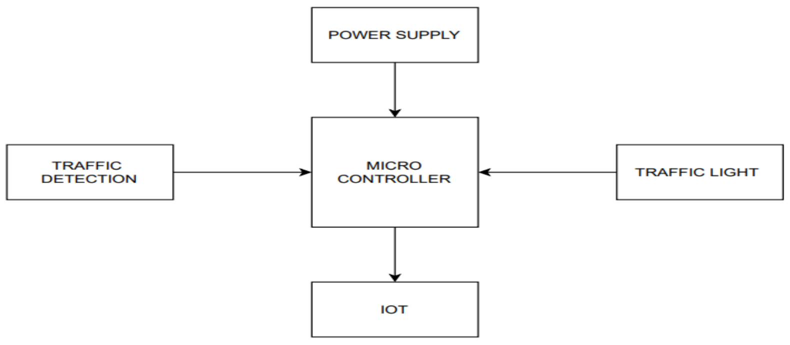


Figure:4.4 Working of Traffic light

CHAPTER 5

IMPLEMENTATION SETUP

5.1 CONVOLUTIONAL NEURAL NETWORK:

The mathematical action known as convolution is carried out via a algorithm known as a "convolutional neural network." A specific kind of linear processing is convolution. Convolutional neural networks, often known as CNNs or ConvNets, are a subset of deep neural networks used in deep learning. They are simple neural networks that incorporate convolutional they all have matrix multiplication in at least one layer. ConvNets have proved effective in distinguishing between objects, people, and traffic. A convolutional neural network consists of an input layer, an output layer, and numerous hidden layers. CNN, or a feed-forward neural network, is a popular tool for classifying and recognizing images. The input is convolved by the Convolutional neural layers, which then provide their output to the following layer. Multilayer perceptron derivatives, or CNNs, are regularized variants. The multilayer perceptron's neurons are all connected to one another in the layer below, making them fully connected networks. Over fitting data is what the "completely linked" network refers to.

5.1.1 CNN ALGORITHM:

CNN for Image Classification:

In order to identify certain patterns in the dataset, image classification requires the extraction of features from a picture. Using an ANN for photo categorization might end up being quite expensive in terms of computation because to the relatively large trainable parameters.

For instance, if we wish to train our conventional ANN on a 50 x 50 image of a cat to determine whether it is a dog or a cat, the trainable parameters are -(50*50) * 100 image pixels multiplied by hidden layer + 100 bias + 2 * 100 output neurons + 2 bias = 2,50,30.

Step 1: Select a Dataset

Choose a dataset that interests you, or make your own picture dataset to address a specific image classification issue. On kaggle.com, selecting a dataset is simple.

This collection includes 12,500 (1000) enhanced traffic photos in JPG format together with labels indicating the traffic category (Class 1, Class 2, or Class 3). There are over 3,000 (450) photographs for each of the 4 categories of traffic, organised into 3 distinct files (according to traffic type). There are three different traffic types: light, medium, and heavy.

High Volume of Traffic (Eosinophil, Lymphocyte, Monocyte, and Neutrophil). The code for importing each library that we would need is shown below.

Step 2: Prepare Dataset for Training

Assigning routes, defining categories (labels), and resizing our photos are all necessary steps in preparing our dataset for training. Image resizing to 200 x 200

Step 3: Create Training Data

An array called training will include the image's pixel values as well as the image's index in the categories list.

Step 4: Shuffle the Dataset

Step 5: Assigning Labels and Features

This shape of both the lists will be used in Classification using the NEURAL NETWORKS.

Step 6: Normalizing X and converting labels to categorical data.

Step 7: Split X and Y for use in CNN

Step 8: Define, compile, and train the CNN Model

Step 9: Accuracy and Score of models

```
function XCOMPRESSCU(*pCurCU)
                M <= FastCUMope(PO,QP)
                if M 4 SPLIT, then
                C2n <=CHECKINTRA (pCurCU)
                 else
                 C2n <= ∞
                end if
                if M != HOMO and Dcur < Dmax then
                Cn <= 0
              for i = 0 to 3 do
          pSubCUi <= pointer to SubCUi
        CN <= CN+ XCompressCU(pSubCUi)
    end for
    else        N <= ∞
    end if
    CHECKBESTMODE(C2N, CN)
     end function
```

5.1.2 DIFFERENT FILTERS AND THEIR EFFECTS:

We may utilise the spatial localization of a given image by using filters, which demand a specific local connection pattern between neurons. Convolution is a mathematical term that describes the pointwise multiplication of two functions to produce a third function. In this instance, our filter is one function, while our image pixel matrix is another.

We obtain the dot product of the two matrices by swiping the filter over the picture. An "Activation Map" or a "Feature Map" is the name of the resultant matrix.

5.1.2.1 Residual Networks (ResNet) in Keras

Because they are more likely to experience disappearing or bursting gradients, very deep neural networks are challenging to train.Because they are more likely to experience disappearing or bursting gradients, very deep neural networks are challenging to train. This issue may be resolved by using a skip connection, which allows Direct passage of an activation unit from one network layer into another deeper layer. Residual networks, often known as ResNets, are built on this.

Residual block

A residual block or identity block is a fundamental component of a ResNet. Simply put, a residual block occurs when a layer's activation in the neural network is accelerated to a deeper layer. The activation of a deeper layer in the network is being added to the activation of a preceding layer, as seen in the graphic above.

This simple tweak allows training much deeper neural networks. The training error of a neural network should, in theory, monotonically decrease as more layers are added. Though, the training error will eventually start to rise for a conventional neural network.This issue does not exist with ResNets. When the network's layers are increased, the training error will continue to decrease. In fact, ResNets have enabled the training of networks with more than 100 layers—up to and including 1000 levels.

Building a ResNet for image classification

Let's now utilise Keras to construct a 50-layer ResNet for picture categorization TensorFlow, CNTK, or Theano may all use the Python-developed Keras high-level neural network API. It was developed with the intention of promoting rapid experimentation. In this case, the backend will be TensorFlow. Of course, feel free to take the whole notebook and import everything before beginning.

Step 1: Define the identity block.

Step 2: Convolution block.

Step 3: Build the model.

Step 4: Training the data.

Step 5: Print the model summary.

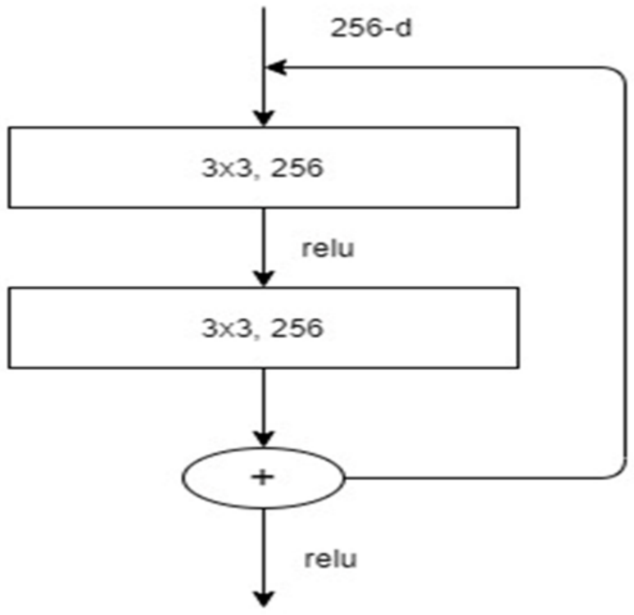


Figure: 5.1 Working of Residual Block

Algorithm 2. Pseudo code of the used preprocessing method.

Input: The raw 1D sensor signal (S) with size of 5625

Output: Graylevel image (Im) with size of 125 x 45

```
1: count = 1;
2:  for i=1 to 125 do
3:  for j=1 to 45 do
4:  Im(i,j) = S(count);
5:  count = count + 1;
6:  end for j
7: end for i
8: Normalize Im by using min-max normalization.
```

4.1.2.2 VGG16 implementation in Keras

The VGG16 architecture of the convolution neural network (CNN) was used to win the 2014 ILSVR (Imagenet) competition. One of the greatest vision model designs now on the market is this one. The main distinguishing characteristic of VGG16 is that it always used the same padding and maxpool layer of a 2x2 filter with a stride 2 and prioritized having convolution layers of a 3x3 filter with a stride 1. Convolution and max pool layers are set up in the same way across the whole design. In the very end, there are two FC (completely connected layers), which are followed by a SoftMax for output. The number 16 in VGG16 stands for the 16 layers with weights. There are over 138 million parameters in this large network.

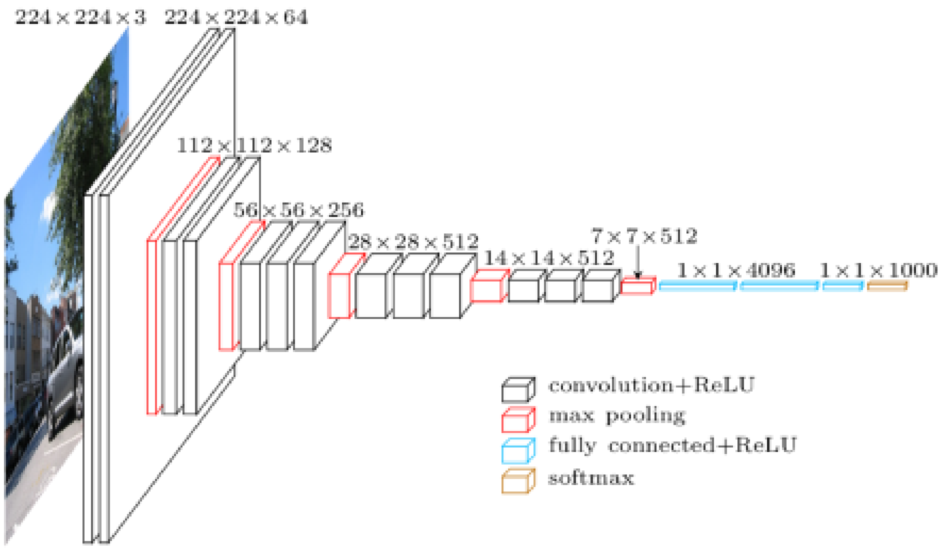


Figure:5.2VGG16 Implementation

This imports all of the libraries needed to implement VGG16. I'll be using the sequential method as I create a sequential model. The layers of the model will be arranged in a consecutive manner, hence the term "sequential model". I have loaded Picture Data Generator from keras preprocessing. The Picture Data Generator's objective is to streamline the model's ability to ingest labelled data. It is a very useful class since it offers a wide range of actions for rescaling, rotating, zooming, and flipping. The best thing about this class is that it doesn't affect the information stored on the disc. This class updates the data in real time as it is being delivered to the model. perform CONV (i, [Bufz])

```
Co <= 0
while Co < Odo
if Co, = 0 and i = 0 then
c<=0
READBIAS()
READKERNEL(Co ~ Co + 31, KC)
end if
|| PPMACConv(K., [Bufr])
|| PREFETCHKERNEL(C, + 32 ~ C, + 63, K.)
C <= C
end while
end function
function CONV(TX, Ty, Ci)
TX <= 0, Ty <= 0,Ci <= 0
while Ty < Y do
while Tx < X do
while Ci< I do
if Tx =0 and Ty = 0 then
 c<=0
READTILE(I  Buf., Tx, Ty, Ci)
end if
|| Convop(Ci, [Buf.)
|| PREFETCHTILE(IBufc., Tx, Ty, Ci + 1

C <= c

end while

end while

end while

end function
```

5.1.2.3 MICE imputation

For imputing missing values, the fancy impute package provides a number of reliable machine learning models. We will utilise the Iterative Imputer, often known as MICE, to impute the missing values in this case. The Iterative Imputer uses aggregates to imput the missing values after performing repeated regressions on random samples of the data. For this imputation, the Data Frame will be used.

- Import IterativeImputer from fancyimpute.
- Copy diabetes to diabetes_mice_imputed.
- Create an IterativeImputer() object and assign it to mice_imputer.
- Impute the diabetes DataFrame.

5.1.3 ADAM Optimizer:

Adaptive moment estimation is a method for gradient descent optimisation. The approach works well for handling complicated issues requiring several variables or pieces of data. It works well and doesn't take up much memory. It naturally mixes the "RMSP" algorithm and the "gradient descent with momentum" approach.

Adam Optimizer Working

The Adam optimizer combines the following two methods of gradient descent:

Momentum:

This method speeds up the gradient descent procedure by employing the gradients' "exponentially weighted average." When averages are utilised, the approach converges more quickly towards the minima.

$$\boldsymbol{w_t + 1 = w_t - \alpha m_t} \qquad (1)$$

Were

$$\boldsymbol{m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta w_t}\right]} \qquad (2)$$

m_t= aggregate of gradients at time t [current] (initially, mt = 0)

m_{t-1} = aggregate of gradients at time t-1 [previous]

w_t = weights at time t

w_{t+1}= weights at time t+1

αt = learning rate at time t

∂L = derivative of Loss Function

∂w_t = derivative of weights at time t

β = Moving average parameter (const, 0.9)

Root Mean Square Propagation (RMSP):

AdaGrad is intended to be improved using the adaptive learning technique known as Root Mean Square Prop, or RMSprop. As contrast to AdaGrad's approach of determining the cumulative sum of squared gradients, it employs the "exponential moving average".

$$\boldsymbol{w_{t+1} = w_t - \frac{\alpha_t}{(v_t + e)\,1/2} * \left[\frac{\delta L}{\delta W_t}\right]} \qquad (3)$$

were,

$$v_t = \beta v_{t-1} + (1-\beta) * \left[\frac{\delta L}{\delta w_t}\right]^2 \qquad (4)$$

w_t = weights at time t

w_{t+1} = weights at time t +1

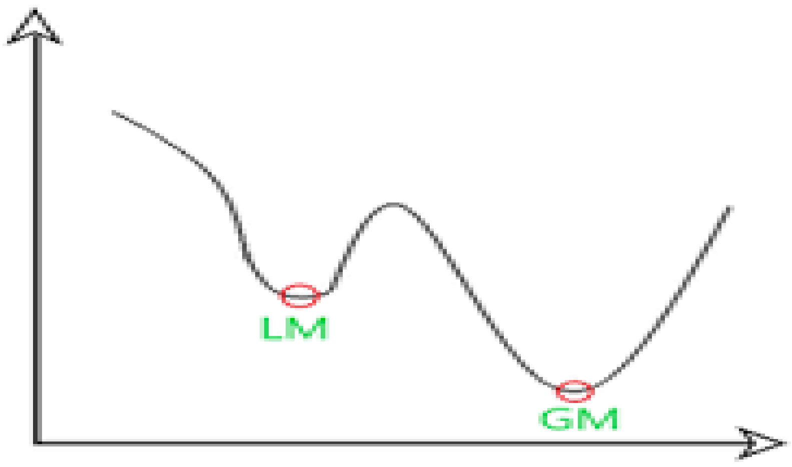
αt = learning rate at time t

∂L = derivative of Loss Function

∂w_t = derivative of weights at time t

v_t = sum of square of past gradients. [i.e sum(∂L / ∂w_t-1)] (initially, v_t = 0)

β = Moving average parameter (const, 0.9)



GM-Global Minimum
LM- Local Minimum

Figure: 5.3 Rate of Gradient Descent

The two approaches have their own strengths, and Adam Optimizer relies on those strengths to provide a gradient descent that is better optimal.

Here, we control the gradient descent rate to overcome the local minima barriers with suitably big steps while achieving the global minimum with the least amount of oscillation feasible (step-size). Hence, combining the benefits of the aforementioned strategies will successfully yield the global minimum.

Mathematical Aspect of Adam Optimizer

$$\boldsymbol{m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2) * \left[\frac{\delta L}{\delta w_t}\right]^2} \qquad (5)$$

Parameters Used:

1. ∈ = a small positive constant to avoid 'division by 0' error when (v_t -> 0). (10-8)

2. β_1 & β_2 = decay rates of average of gradients in the above two methods

(β_1 = 0.9 & β_2 = 0.999)

3. — Learning rate and step size parameter (0.001)

Based on the approaches, mt and vt were both initialised as 0, hence it is noted that they are both "biassed towards 0" since both 1 & 2

By calculating "bias-corrected" mt and vt, this optimizer resolves the issue. In order to avoid large oscillations when close to the global minimum, this is also done to manage the weights as they approach it. The formulae employed are:

$$\boldsymbol{\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \widehat{v}_t = \frac{v_t}{1-\beta_2^t}} \qquad (6)$$

It makes sense that we are modifying the gradient descent after each iteration to keep it consistent and impartial throughout the procedure, hence the name Adam.

We now use the bias-corrected weight parameters (m hat)t and (v hat)t in place of our usual weight parameters mt and vt. Including them in our fundamental equation yields

$$\boldsymbol{w_{t+1=} w_t - \widehat{\widehat{m}_t}\left(\frac{a}{\sqrt{\widehat{v}_t} + e}\right)} \qquad (7)$$

Performance of the Adam Optimizer:

Adam optimizer performs far better than those models and surpasses them in providing an optimised gradient descent based on the benefits of previous models. The graphic below demonstrates how Adam Optimizer outperforms the competition in terms of both performance and training costs (high).

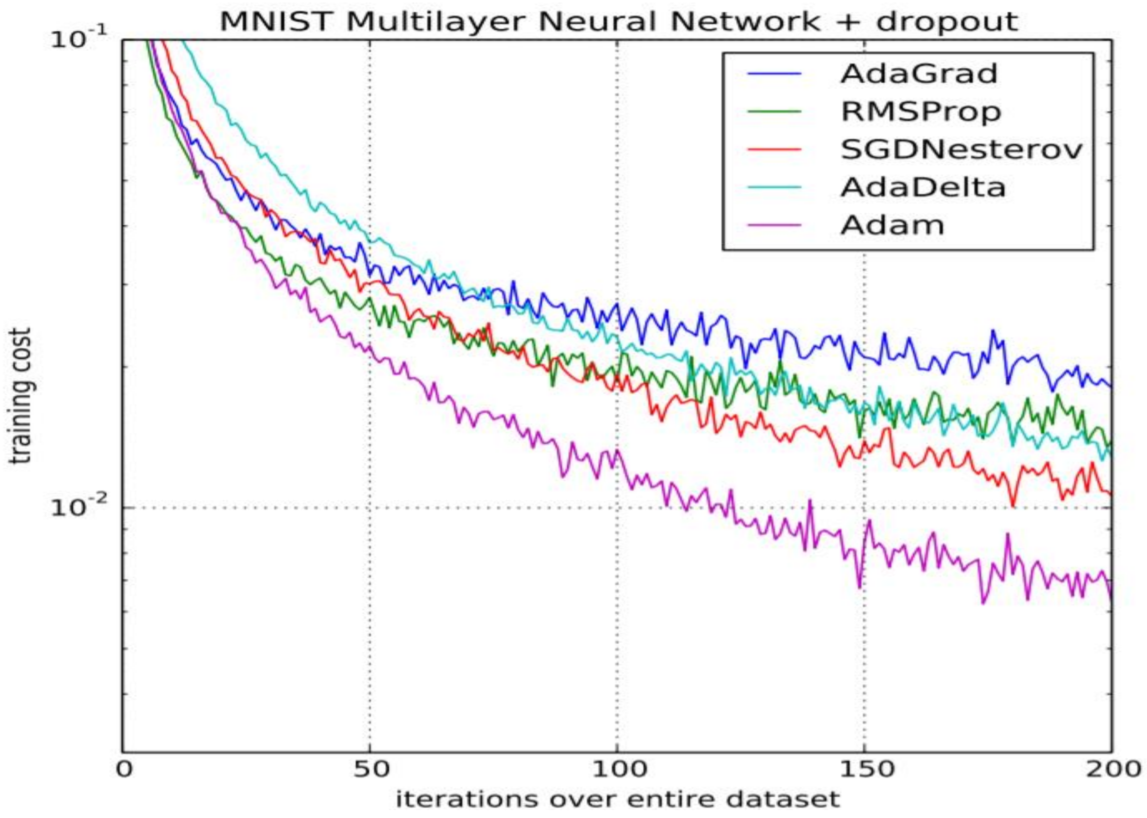


Figure: 5.4 ADAM Optimizer Performance

5.2 SYSTEM SPECIFICATION:

5.2.1 HARDWARE REQUIREMENT

Processor	i3, i5, i10, AMD Processor
RAM	4Gb
Hard Disk	1TB

Microcontroller

ESP32

5.2.2 SOFTWARE REQUIREMENT

Operating System	Windows 8/10
Language	Python (3.10) Version.
Server	GUI

5.2.1 HARDWARE & MODULE DISCRIPTION

5.2.1.1 IOT

A network of physical items that are networked and equipped with electronics, software, sensors, connectors, and connections is known as the Internet of Things (IoT). Every object is, in essence, linked to a tiny, networked computer that allows information to travel to and from it. Lightbulbs, toasters, refrigerators, flowerpots, watches, fans, trains, planes, and anything else in your environment may all be connected to a tiny, networked computer to receive input (particularly object control) or collect data and generate instructional output (typically object status or other sensory data). Computers will thereafter become pervasive embedded computing devices that are connected to the Internet and are visible everywhere. The Internet of things is starting to take off thanks to inexpensive, networkable gadgets and microcontroller chips.

5.2.1.2 MICROCONTROLLER

For those just getting started in the world of embedded systems and microcontrollers, Arduino is a fantastic platform. You may create several projects that are either for fun or are even for sale using a lot of inexpensive sensors and modules. The Internet of Things, sometimes known as IoT, is one of the concepts that emerged as technology developed. It is a platform that is connected to several "things" or devices over the internet to share information. The DIY community mostly concentrates on Home Automation and Smart Home applications, even while commercial and industrial IoT projects have far more complex implementations like Machine Learning, Artificial Intelligence, Wireless Sensor Networks, etc. This brief introduction emphasizes how important Internet connectivity is for all Internet of Things projects, no matter how little, whether they are basic do-it-yourself projects by hobbyists or complex industrial operations. In this case, devices like ESP8266 and ESP32 are used. If you want to provide Wi-Fi connectivity for your projects, the ESP8266 is an excellent option.

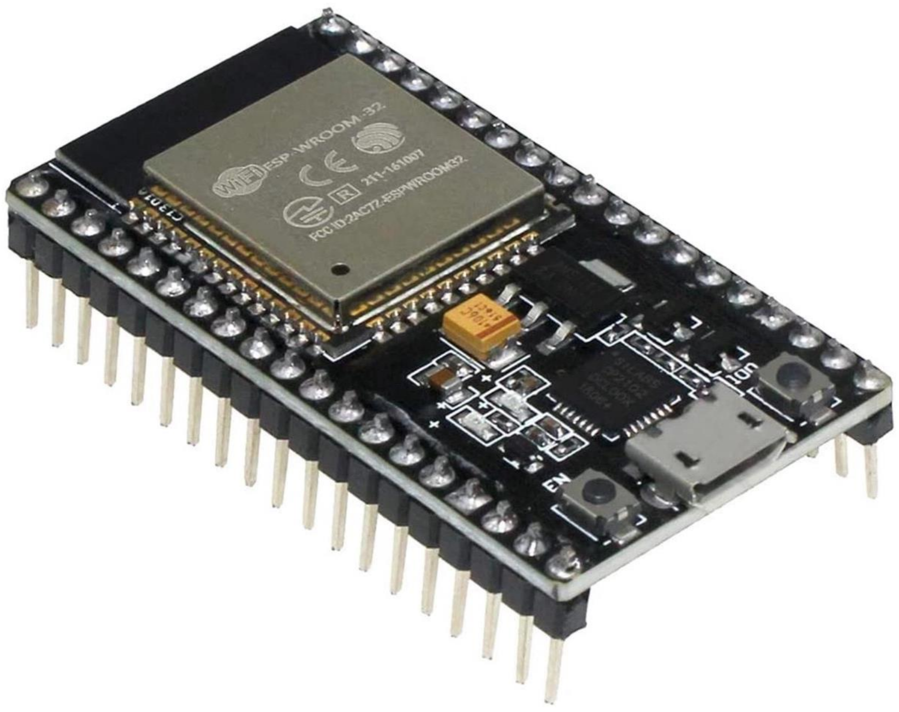


Figure:5.5 Microcontroller

But, ESP32 is the ideal choice if you want to create a full system with Wi-Fi, Bluetooth, high resolution ADCs, DAC, Serial Connectivity, and many other features.

5.2.1.3 ESP3

The well-known ESP8266 System on Chip (SoC) was developed by Espresso Systems, and the less priced ESP32 System on Chip (SoC) Microcontroller is also available. In place of the ESP8266 SoC, Tensilica's 32-bit Xtensa LX6 Microprocessor has Bluetooth and Wi-Fi built in. Versions with a single core and two cores are both offered. RF components including a power amplifier, a low noise receiving amplifier, an antenna switch, filters, and an RF balun are included into the ESP32, just like they are in the ESP8266. This is advantageous. Because few external components are required, it is very easy to build hardware around the ESP32 as a consequence. Espresso's ESP32 delivers a highly integrated Wi-Fi SoC solution to meet customers' continuous demands for consistent performance, a tiny form factor, and low power consumption in the Internet of Things industry. With its full and independent Wi-Fi networking capabilities, the ESP32 may carry out either a standalone application or as the slave to a host MCU.

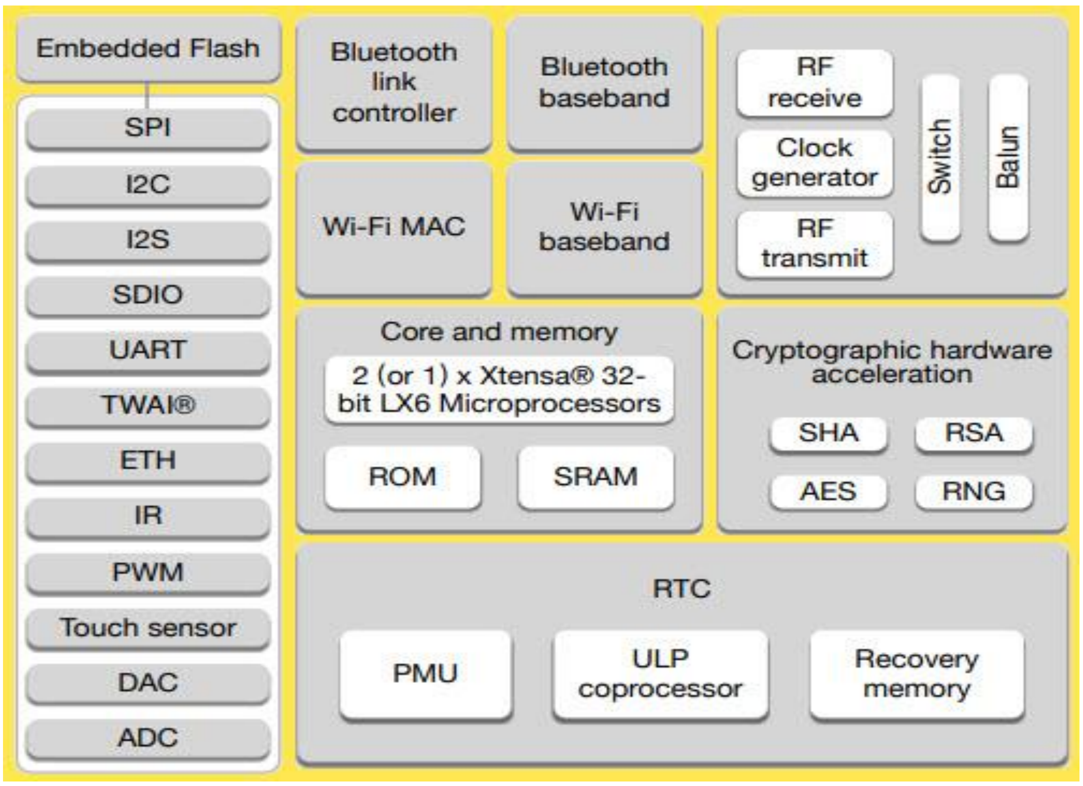


Figure:5.6 Components of ESP-32

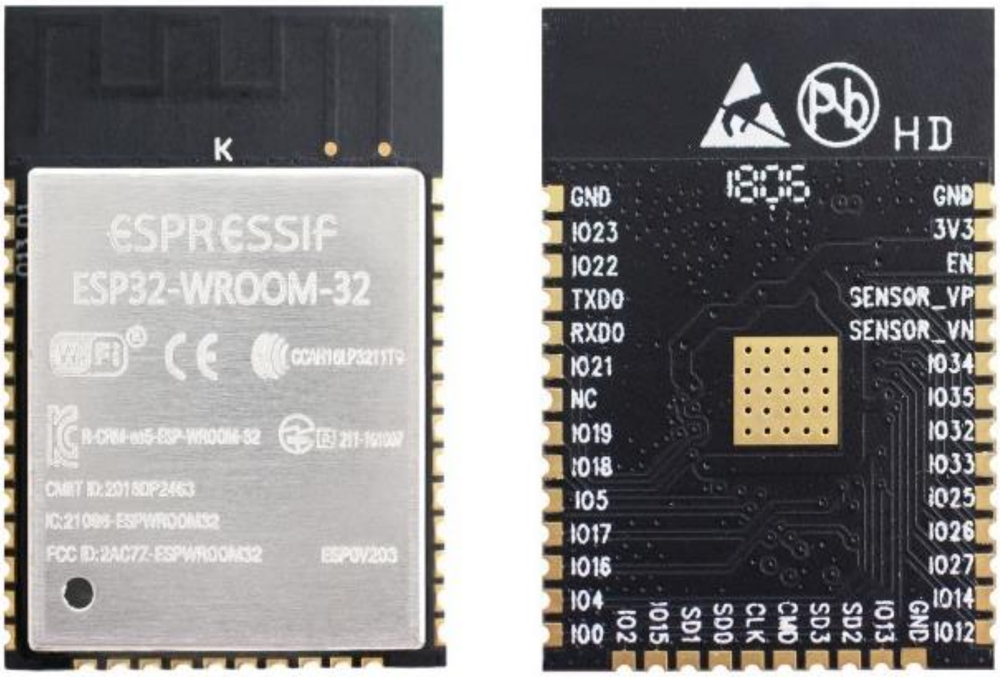


Figure: 5.7 ESP32

If the ESP32 is hosting the software, it starts up instantly from the flash. The integrated high-speed cache helps optimise memory use and boost system performance. Moreover, ESP32 may function as a Wi-Fi adapter in any microcontroller design that makes use of SPI/SDIO, I2C/UART, or other interfaces. The ESP32 comes with antenna switches, RF baluns, power amplifiers, low noise receiver amplifiers, filters, and power management modules. Benefits of the compact design include the tiny PCB size and little external circuitry. In addition to Wi-Fi functionality, the ESP32 has an upgraded L106 Diamond series 32-bit processor from Tensilica and on-chip SRAM. It can link to external sensors and other devices using the GPIOs. The Software Development Kit (SDK) offers sample codes for several applications.

The Smart Connectivity Platform from Espresso Systems enables sophisticated features including adaptive radio biassing for low-power operation, advanced signal processing, surge cancellation, and radio co-existence techniques for common cellular, Bluetooth, DDR, LVDS, and LCD interference reduction (ESCP). For energy-saving objectives, these characteristics also allow for quick transitions between the sleep and wake up modes.

Channel Frequencies:

- The following channels are supported by the RF transceiver in accordance with IEEE802.11b/g/n specifications.

GHz Receiver:

- The 2.4 GHz receiver uses two high-resolution, high-speed ADCs to down-convert RF signals into quadrature baseband signals before converting them to the digital realm. RF filters, automatic gain control (AGC), DC offset cancellation circuits, and baseband filters are included inside the ESP32 to respond to changing signal channel circumstances.

GHz Transmitter:

- The 2.4 GHz transmitter upconverts the quadrature baseband signals to 2.4 GHz and uses a powerful CMOS power amplifier to drive the antenna. The power amplifier's linearity is further enhanced by the function of digital calibration, resulting in state-of-the-art performance that can achieve +19.5 dBm.average power for 802.11b transmission and +16 dBm for 802.11n transmission.

- Additional calibrations are integrated to offset any imperfections of the radio, such as:

- Carrier leakage

- I/Q phase matching

- Baseband nonlinearities
- These built-in calibration functions reduce the product test time and make the test equipment unnecessary.

FEATURES

- Power Supply: DC +12v 1Amp.
- Auto data updating: 30sec.
- Digital Output port Pins: +5V DC
- Message Format: *message or Data # (Start with * and End with #)
- Provided with 3 links.
- Data updating to a specific web site.
- Device controlling web site.
- Data updating to a social network.

APPLICATIONS

- Online Traffic monitoring
- Online Health monitoring
- Real time Transport and Logistics monitoring

5.2.1.4 LED

Light-emitting diodes (LEDs) are a common source of illumination for electrical devices. It may be used for many different things, such as mobile phones and enormous billboards. They are frequently employed in devices that show time and numerous types of data. A light-emitting diode (LED) is a semiconductor device that generates light when an electric current flows through it. An LED produces light when current passes across it by recombining its electrons and holes. LEDs restrict the direction that electricity may travel in and only allow it to move ahead. An LED's hue is influenced by the material used in the semiconducting element. The two major materials used in LEDs are indium gallium nitride alloys and aluminum gallium indium phosphide alloys. Aluminum alloys are utilized to create red, orange, and yellow light, whereas indium alloys are used to create green, blue, and white light. As the compositions of these alloys are slightly changed, the color of the light that is emitted changes. Several sectors, including optical communication, remote-controlled operations, robotics, alarm, and security systems, among others, employ LEDs. It is employed in several applications due to its robustness, low power requirements, short reaction time, and quick switching capabilities. Among the criteria that LED upholds are the following:

- Used for TV backlighting.
- Used in displays.
- Used in Automotives.
- LEDs used in the dimming of lights.

Figure: 5.8 LED Traffic signals

Due to their high efficiency, low power consumption, and long lifespan, LEDs have already replaced traditional incandescent lights in traffic signal applications. With LED traffic signals, high power and brightness LEDs are connected in parallel and series to form an LED cluster. To obtain a specific luminous distribution intensity, a second optical design (lenses) is necessary. If traffic lights are AC-powered, a circuit that converts AC power to DC power appropriate for an LED driver is required to run an LED cluster or string. MCU is used to combine control over the LED driver and produce precise traffic signals.

5.2.2 SOFTWARE & MODULE DISCRIPTION

5.2.2.1 PYTHON

Python is a high-level object-oriented programming language that was created by Guido van Rossum. It is also known as a general-purpose programming language since it is used in almost every sector we can think of, including those that are listed below.

- Internet Development
- Software Development
- Game Development
- AI & ML
- Data Analytics

Each programming language has a feature or use case that is unique to a certain industry. For instance, JavaScript is the most popular language used by web developers because it gives the developer the opportunity to manage applications utilizing several frameworks, including react, Vue, and angular, which are used to construct aesthetically pleasing User Interfaces. Python is consequently seen as having general-purpose, which indicates that it is widely utilized across all industries. This is because it is scalable and reasonably easy to understand, which helps to how quickly it is developing. Because of Python's syntax's resemblance to the English language, programmers may create programmers with less lines of code. As it is open source, several libraries are accessible, which facilitates developers' work and boost output. They can only focus on business logic and its difficult capabilities in the digital era when information is available in vast data sets.

Applications of Python Programming:

1) **Web Development:** Python provides a variety of web development frameworks, including Django, Pyramid, and Flask. This framework has a reputation for being safe, adaptable, and scalable.

2) **Game Development:** Two Python libraries used for game creation are PySoy and PyGame.

3) **Artificial intelligence and machine learning:** (AI/ML) applications may be created using a variety of open-source libraries.

CHAPTER 6

RESULT AND INFERENCES

The results obtained are based on the training and building the CNN model. An application in Fig 6.1 is built where the image from the dataset is uploaded. The application makes use of the model to classify the image according to the density of the traffic as no traffic, low, medium, and high traffic. The uploaded image is preprocessed which consists of several stages such as image capture, edge detection, and traffic detection classification. To create a suitable and clean image, we will do many operations which include grayscale conversion, filtering, sharpening, smoothing, edging, and image segmentation. These images are processed using Python Image Library (PIL) and Numpy libraries-supporting large multidimensional arrays and matrices. After the Pre-Processing the image is classified according to the density of the vehicles on the road.

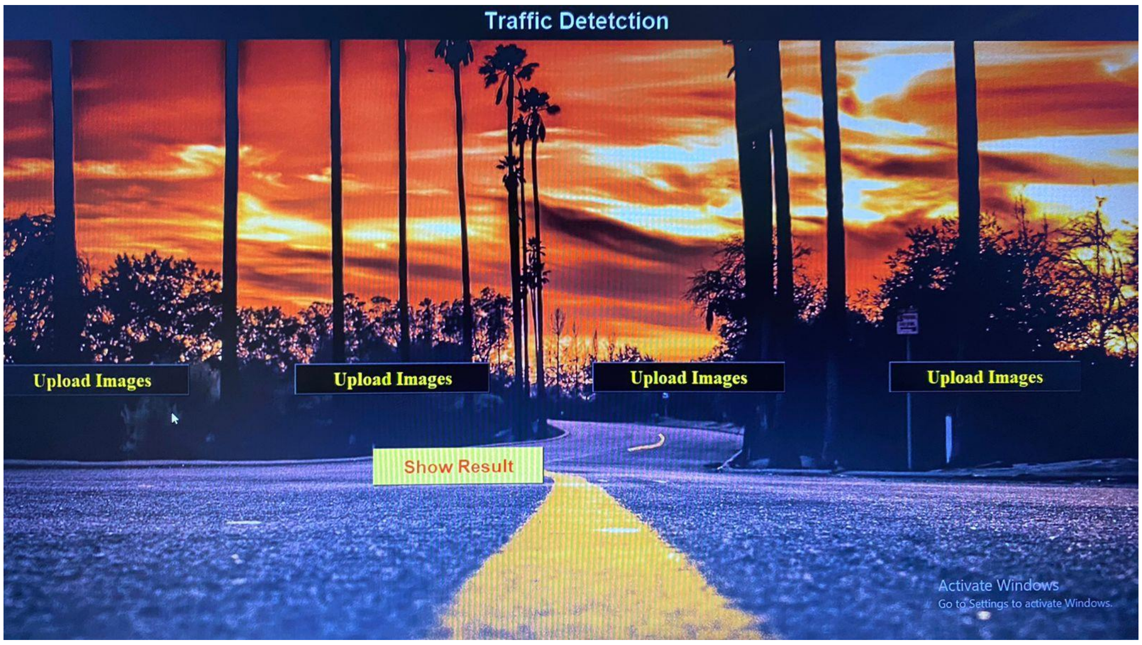


Figure: 6.1 Application to detect traffic

The density on each lane is calculated and determined individually. The classified image is sent to the microcontroller through the Wi-Fi module which is connected to the laptop.

The microcontroller then assesses the volume of traffic, assigns timings for each lane, and sends the appropriate signal to the red, yellow, and green LEDs that serve as traffic light indicators. The configuration of the microcontroller is shown in Fig. 5. The densities on Roads 1, 2, 3, and 4 are, respectively, low, medium, no traffic, and high, as seen in the image. The identical values reported by the sensors are supported by the matching values shown on the Python terminal.Therefore, In Fig 6.2 the images with different density of vehicle on the road are uploaded.

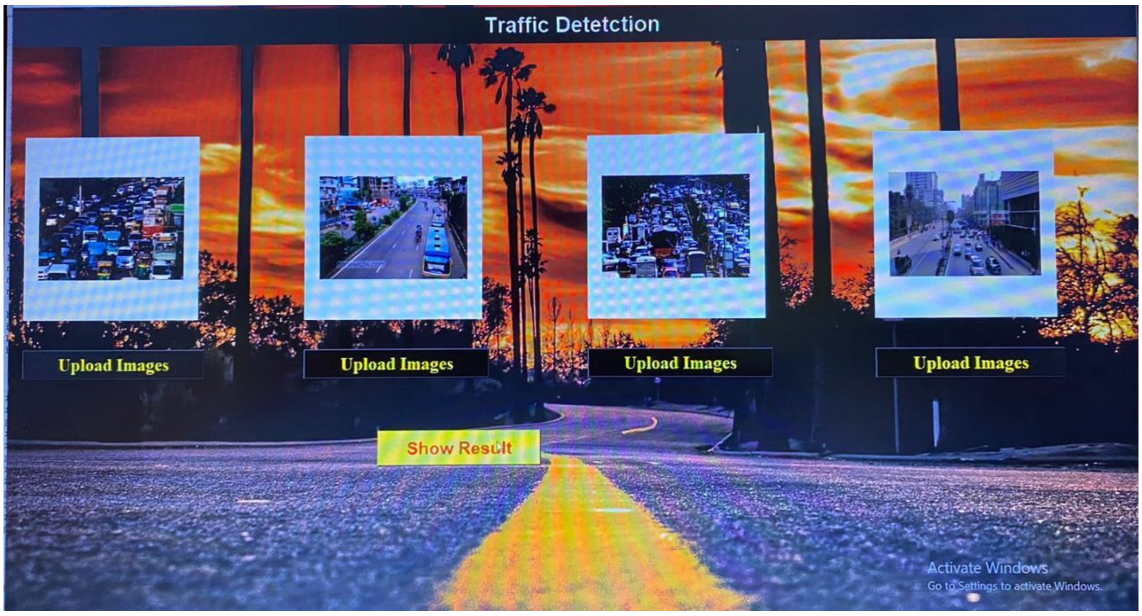


Figure:6.2 Road lanes with traffic

In the above Fig 6.2 the first image is lane1, second image represents the traffic in lane 2 and so on. Here, the images are preprocessed and classified in the backend using the Python libraries and models. The output is classified as no traffic, low, medium, and high traffic and sends the classified result to the microcontroller. According to the uploaded images, first the traffic light in the lane 1 changes, despite the density in other lanes. After the change in the signal in lane 1 the density is considered for the change of the signal in all other lanes.

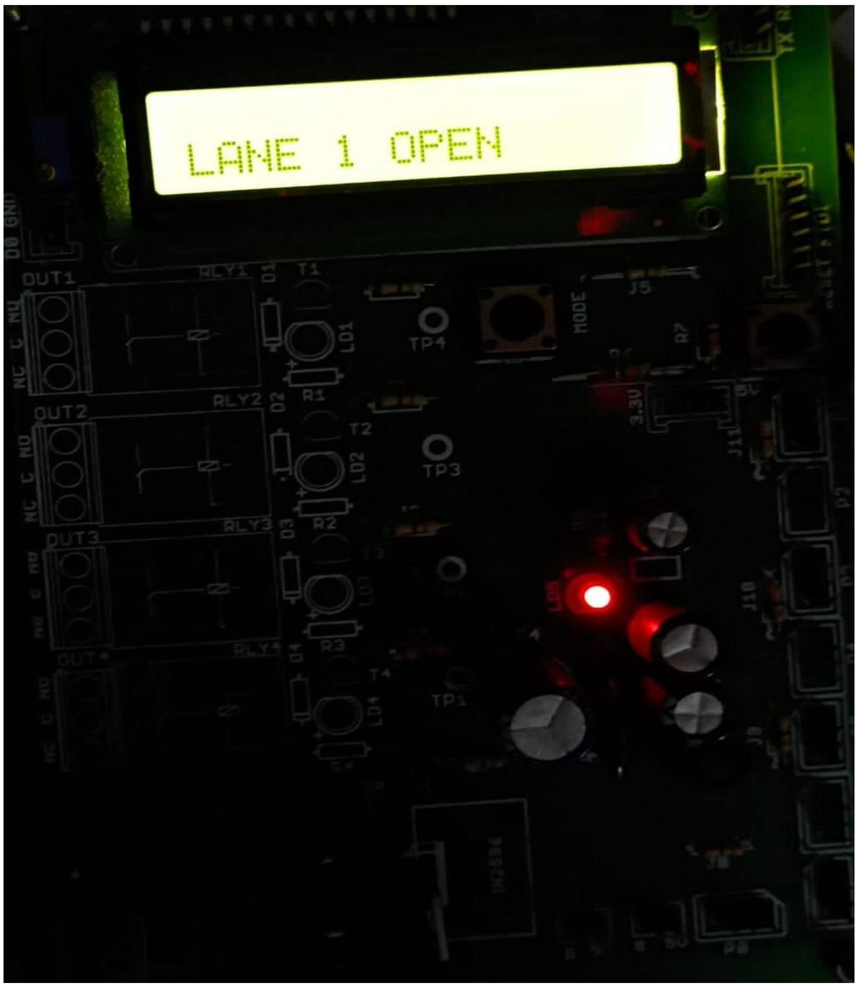


Figure:6.3.1 Lane 1

Therefore, Fig 6.3.2 shows the change in signal from lane 1 to lane 2. The signal in lane 2 has low density and green signal stays for only few minutes. After lane 1, lane 2 opens, then lane 4 opens and finally lane 3 opens.

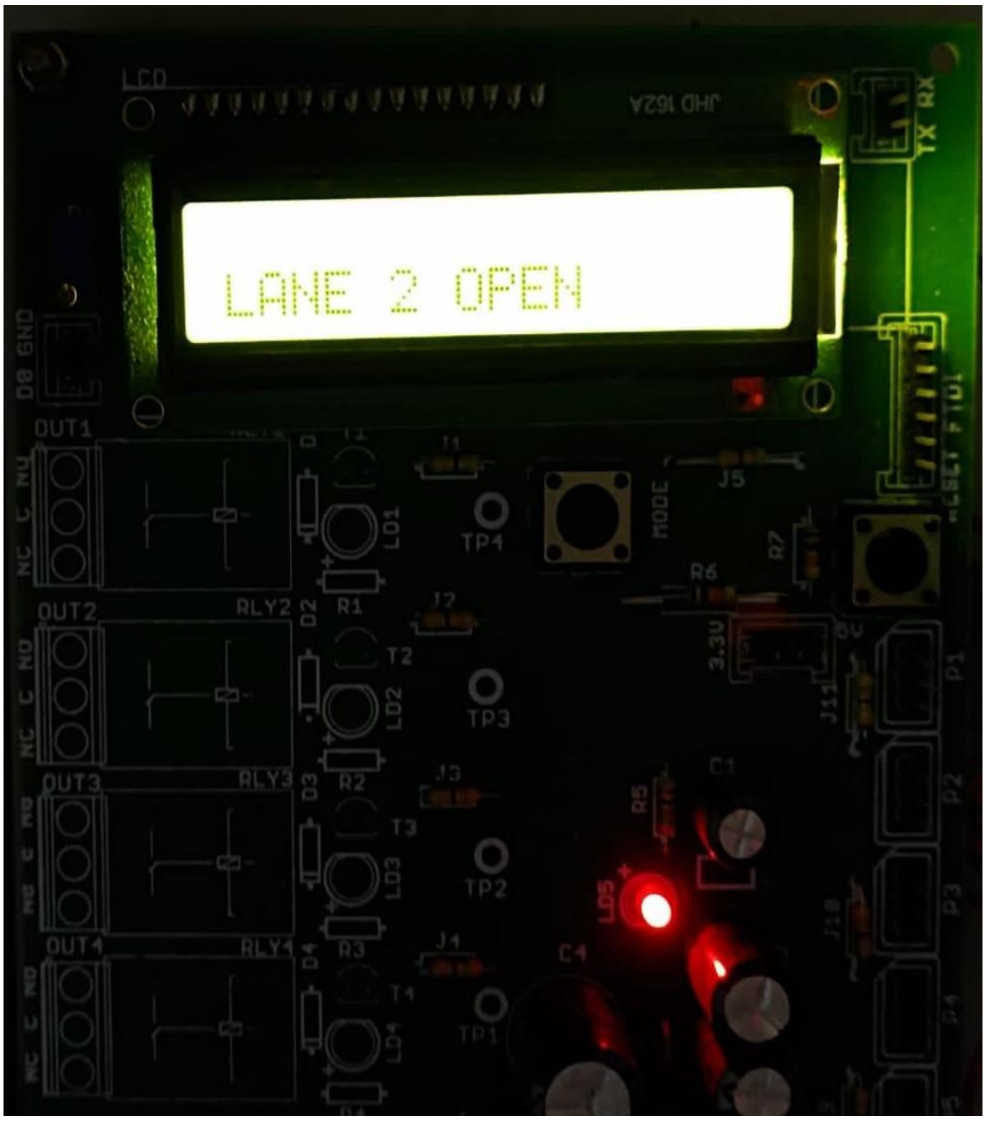


Figure:6.3.2 lane 2

Fig6.3.3 shows the lane 4, The density in lane 4 is low compared to lane 3, so lane 4 opens and the lights change according to the traffic and the timings. Fig 6.3.4 shows the change from lane 4 to lane 3. In the last lane 3 opens and the green signal lasts of more time as there is a huge traffic density.

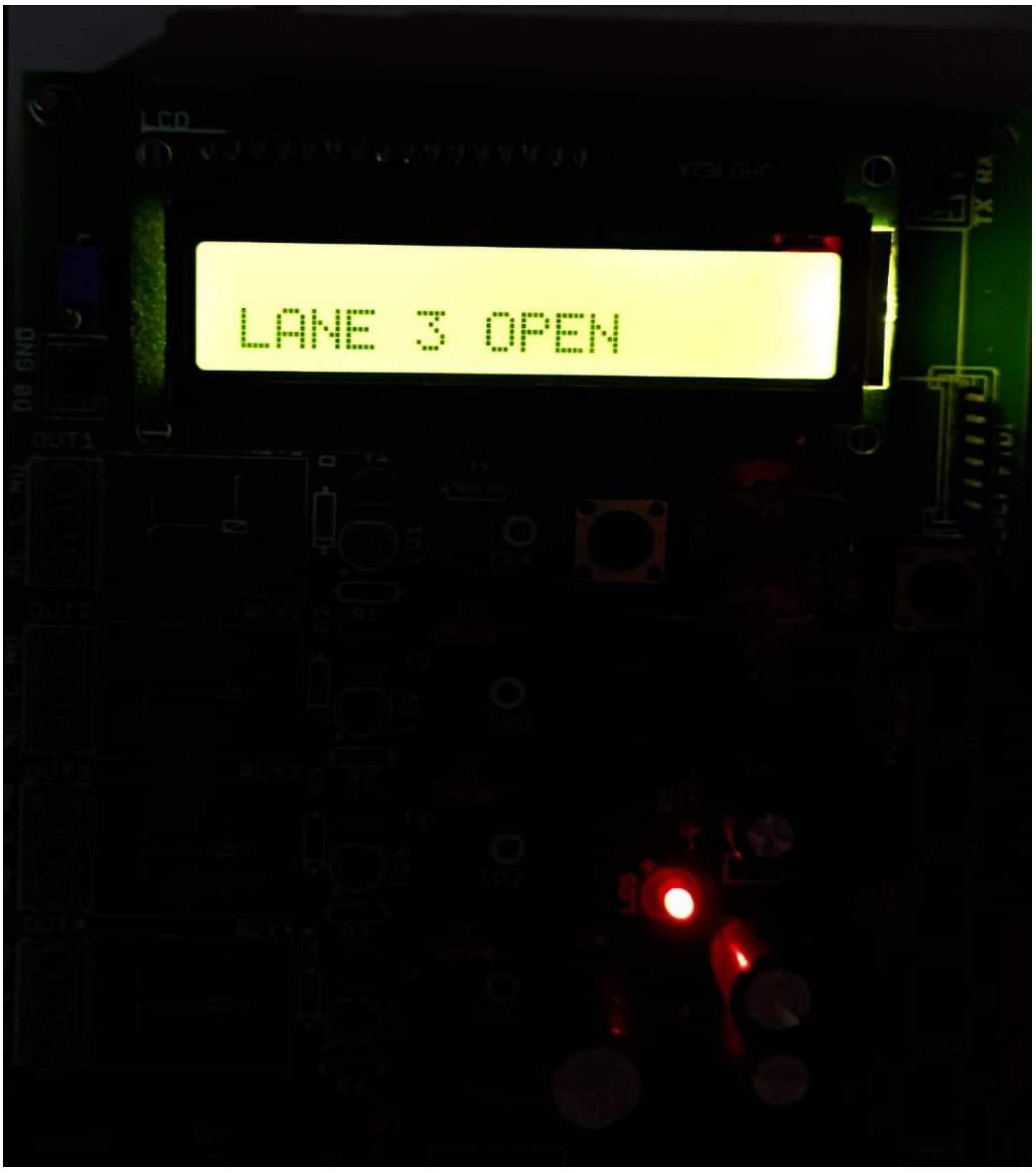


Figure:6.3.3 Lane 3

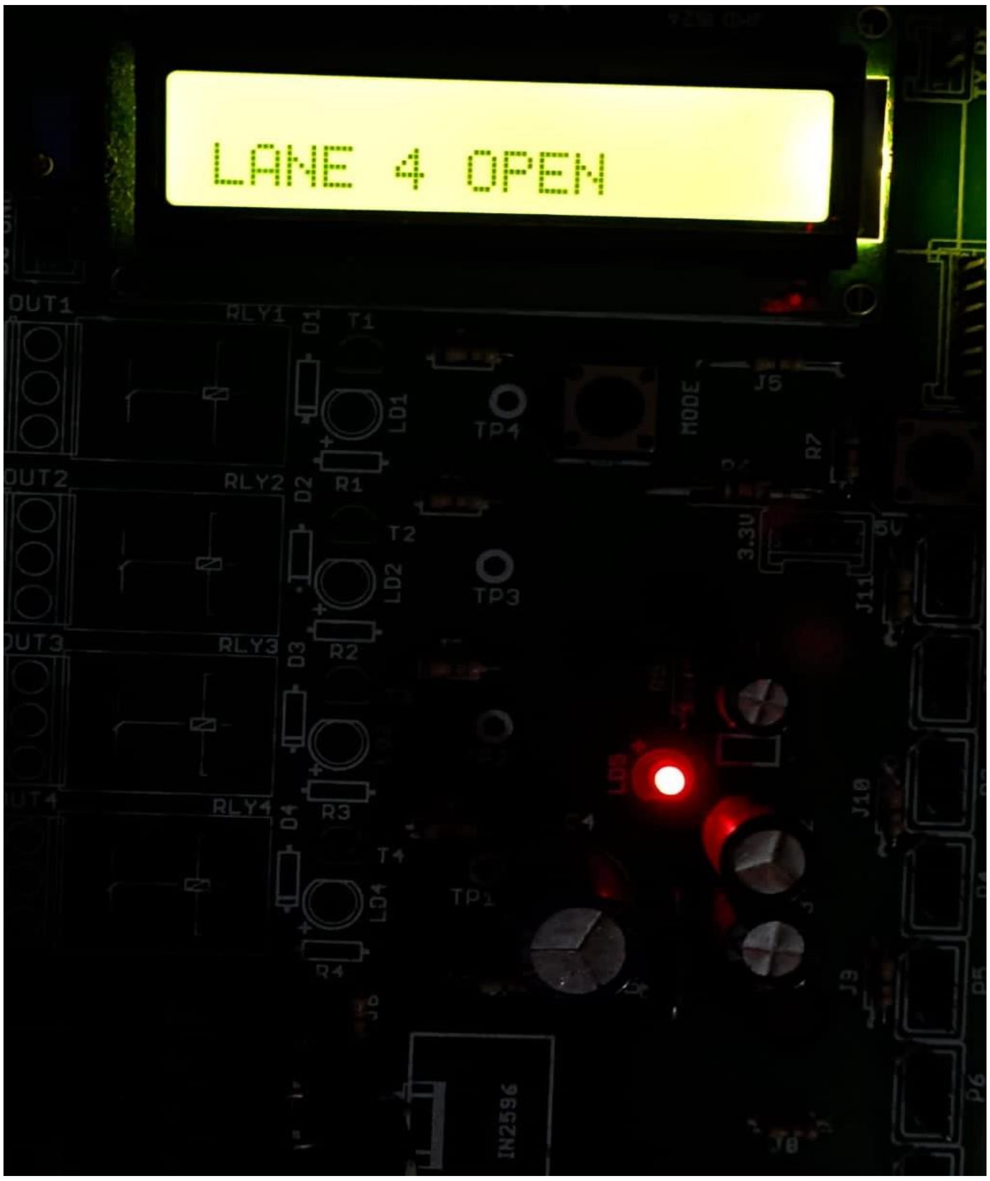


Figure:6.3.4 Lane 4

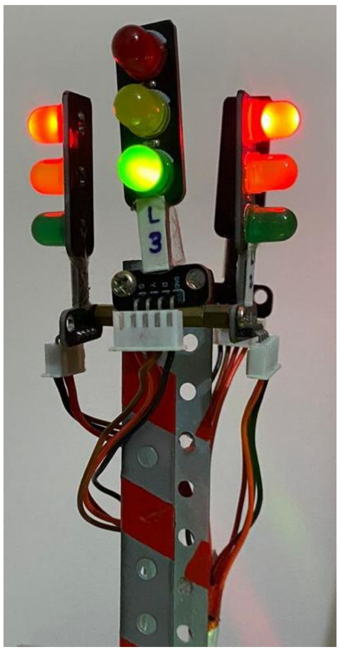


Figure:6.4 Traffic Light

Because of the aforementioned findings, it can be concluded that an estimated level of traffic that is comparable to real-time values of traffic may be determined using real-time photos acquired and image processing algorithms. Depending on the actual traffic density, this information may be utilized to evaluate and manage the traffic signals in real time. This will also lessen the negative consequences of traffic congestion while saving time.

CHAPTER 7
CONCLUSION AND FUTURE WORK

In this study, we used transfer learning to create a CNN model for autonomous traffic recognition using road photographs. In transfer learning, weights from networks that have been trained on millions of data are employed. The proposed study employs four distinct transfer learning models (ADAM, SGD, RMSprop), each with a unique optimizer. On datasets with the most road photos available right now, extensive trials were conducted. This model uses the SoftMax layer, three dense layers, and transfer learning to extract the features for classification. The recommended deep TL models are shown to train quickly using the Adam optimizer, and the overfitting problem is avoided using the dropout method. The system's performance can still be improved in the future by leveraging bigger datasets and other deep learning methods.

7.1 FUTURE SCOPE

Future editions of the system will frequently include additional traffic management-related capabilities by utilising techniques like deep learning, artificial neural networks, and even big data. The users can utilise this method to look for the path that would make it the simplest for them to reach their goal. Consumers can make use of technology to assist them in their search preferences and pick the simplest option in a less crowded area. Road traffic bottleneck forecasting has already used a variety of forecasting techniques. Even if there is still room for improvement in the accuracy of the congestion prediction, there are more methods that offer precise forecasts.

REFERENCE

[1] B. Hussain, M. K. Afzal, S. Ahmad and A. M. Mostafa. (2021). Intelligent TFP Using Optimized GRU Model. IEEE Access, 9(1), 100736-100746

[2] B. K. P. Horn and B. G. Schunck, Horn Schunck Determining Optical_Flow, vol. 17.Issue1-3,August 1981.

[3] Guo, K., Hu, Y., Qian, Z., Liu, H., Zhang, K., Sun, Y., & Yin, B. (2020). Optimized graph convolution recurrent neural network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 22(2), 1138-1149.

[4] J. Redmon, S. Divvala, R. Girshick and A. Farhadi (2015), You Only Look Once: Unified Real-Time Object Detection.

[5] Jia, T., & Yan, P. (2020). Predicting citywide road traffic flow using deep spatiotemporal neural networks. IEEE Transactions on Intelligent Transportation Systems, 22(5), 3101-3111.

[6] Kumar, B. R., Chikkakrishna, N. K., & Tallam, T. (2020). Short Term Predictions of Traffic Flow Characteristics using ML Techniques. In 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA) 4(1),1504-1508.

[7] Ma, Q., Huang, G. H., & Ullah, S. (2020). A Multi-Parameter Chaotic Fusion Approach for Traffic Flow Forecasting. IEEE Access, 8(1), 222774-222781.

[8] Peng, H., Du, B., Liu, M., Liu, M., Ji, S., Wang, S., ... & He, L. (2021). Dynamic graph convolutional network for long-term TFP with reinforcement learning. Information Sciences, 578, (1) 401-416.

[9] Qu, W., Li, J., Yang, L., Li, D., Liu, S., Zhao, Q., & Qi, Y. (2020). Short-term intersection Traffic flow forecasting. Sustainability, 12(19), 1-13.

[10] S. Ren, K. He, R. Girshick and J. Sun (2017), "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137-1149.

[11] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet (2017): A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation", IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 12, pp. 2481-2495.

[12] Wang, Z., Su, X., & Ding, Z. (2020). Long-term traffic prediction based on lstm encoder-decoder architecture. IEEE Transactions on Intelligent Transportation Systems, 22(10), 6561-6571.

[13] Xu, H., & Jiang, C. (2020). Deep belief network-based support vector regression method for traffic flow forecasting. Neural Computing and Applications, 32(7), 2027-2036.

[14] Y. Zhou, L. Liu, L. Shao and M. Mellor, "DAVE (2016): A unified framework for fast vehicle detection and annotation", Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9906 LNCS, pp. 278-293.

[15] Zhao, F., Zeng, G. Q., & Lu, K. D. (2019). EnLSTM-WPEO: Short-term TFP by ensemble LSTM, NNCT weight integration, and population extremal optimization. IEEE Transactions on Vehicular Technology, 69(1), 101-113.

[16] L. F. P. de Oliveira, L. T. Manera and P. D. G. D. Luz, "Development of a Smart Traffic Light Control System With Real-Time Monitoring," in *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3384-3393, 1 March1, 2021, doi: 10.1109/JIOT.2020.3022392.

[17] M. M. Gandhi, D. S. Solanki, R. S. Daptardar and N. S. Baloorkar, "Smart Control of Traffic Light Using Artificial Intelligence," *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, India, 2020, pp. 1-6, doi: 10.1109/ICRAIE51050.2020.9358334.

[18] W. A. C. J. K. Chandrasekara, R. M. K. T. Rathnayaka and L. L. G. Chathuranga, "A Real-Time Density-Based Traffic Signal Control System," *2020 5th International Conference on Information Technology Research (ICITR)*, Moratuwa, Sri Lanka, 2020, pp. 1-6, doi: 10.1109/ICITR51448.2020.9310906.

[19] A. M. H. Yusuf and R. Yusuf, "Adaptive Traffic Light Controller Simulation for Traffic Management," *2020 6th International Conference on Interactive Digital Media (ICIDM)*, Bandung, Indonesia, 2020, pp. 1-5, doi: 10.1109/ICIDM51048.2020.9339649.

[20] M. Sutharsan, S. Rajakaruna, S. Y. Jayaweera, J. A. C. M. Jayaweera and S. Thayaparan, "Vision-Based Adaptive Traffic Light Controller for Single Intersection," *2020 5th International Conference on Information Technology Research (ICITR)*, Moratuwa, Sri Lanka, 2020, pp. 1-6, doi: 10.1109/ICITR51448.2020.9310872.

[21] López-Bonilla, O.R.; García-Guerrero,E.E.; Tlelo-Cuautle, E.; López-Mancilla, D.; Hernández-Mejía, C.;Inzunza-González, E. Traffic FlowPrediction for Smart Traffic Lights Using Machine Learning Algorithms.Technologies 2022.

[22] M. Ahmed, S. Masood, M. Ahmad and A. A. Abd El-Latif, "Intelligent Driver Drowsiness Detection for Traffic Safety Based on Multi CNN Deep Model and Facial Subsampling," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19743-19752, Oct. 2022, doi: 10.1109/TITS.2021.3134222.

APPENDIX A1: SOURCE CODE

CODING:

Class1.py:

```
import tensorflow as tf

from tensorflow import keras

from keras_preprocessing.image import ImageDataGenerator

from keras_preprocessing import image

import numpy as np

import easygui

import os

import serial

import absl.logging

absl.logging.set_verbosity(absl.logging.ERROR)

print(tf.__version__)

train_datagen = ImageDataGenerator(

rescale=1./255,

shear_range=0.2,

zoom_range=0.2,

horizontal_flip=True)
```

```
train_set = train_datagen.flow_from_directory(

'Class1/training_set',

target_size=(64, 64),

batch_size=32,

class_mode='binary')

test_datagen = ImageDataGenerator(rescale=1./255)

test_set = test_datagen.flow_from_directory(

'Class1/test_set',

target_size=(64, 64),

batch_size=32,

class_mode='binary')

print(test_set)

#--------------------- Building CNN --------------------#

cnn = tf.keras.models.Sequential()

cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu',
input_shape=[64,64,3]))

cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))

cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu'))

cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))

cnn.add(tf.keras.layers.Flatten()
cnn.add(tf.keras.layers.Dense(units = 128, activation = 'relu'))

cnn.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))

#-------------------- Training the CNN -----------------#

cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

cnn.fit(x = train_set, validation_data = test_set, epochs = 25)

cnn.save('model/save1',overwrite=True,

include_optimizer=True,

save_format=None,

signatures=None,

options=None,

save_traces=True,)

cnn.save('model/Class1/model_Class1.h5')

a="continue"

while a=="continue":

image11 = easygui.fileopenbox()

test_image2 = image.load_img(image11, target_size = (64, 64))

test_image2 = image.img_to_array(test_image2)

test_image2 = np.expand_dims(test_image2, axis = 0)

result2 = cnn.predict(test_image2) print(result2)
```

```
if result2[0][0] == 1:

prediction2 = 'No emergency vehicle detected'

else:

prediction2 = 'emergency vehicle detected'

print(prediction2)

print("Type your ans")

b=str(input("continue or  exit ; "))

a=b.lower()
```

Class2.py:

```
import tensorflow as tf

from keras_preprocessing.image import ImageDataGenerator

from keras_preprocessing import image

import numpy as np

import easygui

import os

import serial

import absl.logging

absl.logging.set_verbosity(absl.logging.ERROR)

print (tf. __version__)

train_datagen = ImageDataGenerator (

rescale=1./255,

shear_range=0.2
```

```
zoom_range=0.2,

horizontal_flip=True)

train_set = train_datagen.flow_from_directory(

'Class2/training_set',

target_size=(64, 64),

batch_size=32,

class_mode='binary')

test_datagen = ImageDataGenerator(rescale=1./255)

test_set = test_datagen.flow_from_directory(

'Class2/test_set',

target_size=(64, 64),

batch_size=32,

class_mode='binary')

print(test_set)

#--------------------- Building CNN --------------------#

cnn = tf.keras.models.Sequential()

cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu',
input_shape=[64,64,3]))

cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))

cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu'))

cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))

cnn.add(tf.keras.layers.Flatten()
cnn.add(tf.keras.layers.Dense(units = 128, activation = 'relu'))

cnn.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))

#-------------------- Training the CNN -------------------#

cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

cnn.fit(x = train_set, validation_data = test_set, epochs = 25)

cnn.save('model/save1',overwrite=True,

include_optimizer=True,

save_format=None,

signatures=None,

options=None,

save_traces=True,)

cnn.save('model/Class2/model_Class2.h5')

a="continue"

while a=="continue":

image11 = easygui.fileopenbox()

test_image2 = image.load_img(image11, target_size = (64, 64))

test_image2 = image.img_to_array(test_image2)

test_image2 = np.expand_dims(test_image2, axis = 0)

result2 = cnn.predict(test_image2)
```

```
print(result2)

if result2[0][0] == 1:

prediction2 = 'Medium or High Traffic'

else:

prediction2 = 'Low Traffic'

print(prediction2)

print("Type your ans")

b=str(input("continue or  exit ; "))

a=b.lower()
```

main.py:

```
import tensorflow as tf

import sys

from keras_preprocessing.image import ImageDataGenerator

from keras_preprocessing import image

import numpy as np

import easygui

from keras.models import load_model

import os

import serial

import tkinter as tk

from tkinter import * from tkinter import filedialog
from tkinter.filedialog import askopenfile

from PIL import Image, ImageTk

my_w = tk.Tk()

my_w.geometry('1350x710+0+10')

my_w.title('www.trafficdetetction.com')

my_font1=('times', 18, 'bold')

bg = PhotoImage(file='background.png')

bgLabel = Label(my_w, image=bg)

bgLabel.place(x=0, y=0)

l1 = tk.Label(my_w,text='Upload Files & get
results',width=30,font=my_font1,bg='seagreen2',

fg='black',)

l1.place(x=420, y=120, width=300)

b1 = tk.Button(my_w, text='Upload Images',

width=20,command = lambda:traffic_type(), activebackground='seagreen1',
bg='seagreen2',fg='black')

b1.place(x=455,y=620, width=240, height=40)

print(tf.__version__)

titleLabel = Label(my_w, text='Traffic Detetctor', font=('italic', 27, 'bold'), bg='black',

fg='seagreen2',anchor=CENTER )

titleLabel.place(x=5, y=10,width=1100, height=80)

model1 = load_model('model/Class1/model_Class1.h5'
model2 = load_model('model/Class2/model_Class2.h5')

model3 = load_model('model/Class3/model_Class3.h5')

def result():

filename =upload_file()

test_image2=image.load_img(filename,target_size(64, 64))

test_image2 = image.img_to_array(test_image2)

test_image2 = np.expand_dims(test_image2, axis = 0)

# cnn prediction on the test image

result1 = model1.predict(test_image2)

print(result1)

result2 = model2.predict(test_image2)

print(result2)

result3 = model3.predict(test_image2)

print(result3)

if result1[0][0] == 1:

if result2[0][0] == 1:

if  result3[0][0] == 1:

prediction2='Traffic Only Predicted'

else:

prediction2='Traffic Only Predicted'

else:

prediction2='Traffic Only Predicted'
```

```
else:
prediction2 = 'Emergency Vehicle Predicted'
print(prediction2)
prediction=prediction2
l2 = tk.Label(my_w,text="Result :  "+prediction,width=30,font=my_font1,bg='black',
fg='seagreen2',)
l2.place(x=115, y=450, width=400,height=55)
return filename
def traffic_type():
image11 =result()
test_image2 = image.load_img(image11, target_size = (64, 64))
test_image2 = image.img_to_array(test_image2)
test_image2 = np.expand_dims(test_image2, axis = 0)
# cnn prediction on the test image
result1 = model1.predict(test_image2)
print(result1)
result2 = model2.predict(test_image2)
print(result2)
result3 = model3.predict(test_image2)
print(result3)
if result1[0][0] == 1:
```

```
if result2[0][0]==0:

prediction2="Low Traffic"

else:

if result3[0][0]==0:

prediction2="Medium Traffic"

else:

prediction2prediction2="Ambulance"
print(prediction2)

prediction=prediction2

l3 = tk.Label(my_w,text="Type  :  "+prediction,width=30,font=my_font1,bg='black',

fg='seagreen2',)

l3.place(x=475, y=515, width=440,height=55)

return prediction2

def upload_file():

f_types = [('JPG Files', '*.JPG'),

('PNG Files','*.png')]

filename = tk.filedialog.askopenfilename(multiple=False,filetypes=f_types)

img=Image.open(filename) # read the image file

img=img.resize((200,140)) # new width & height

img=ImageTk.PhotoImage()
```

APPENDIX B: SNAPSHOT

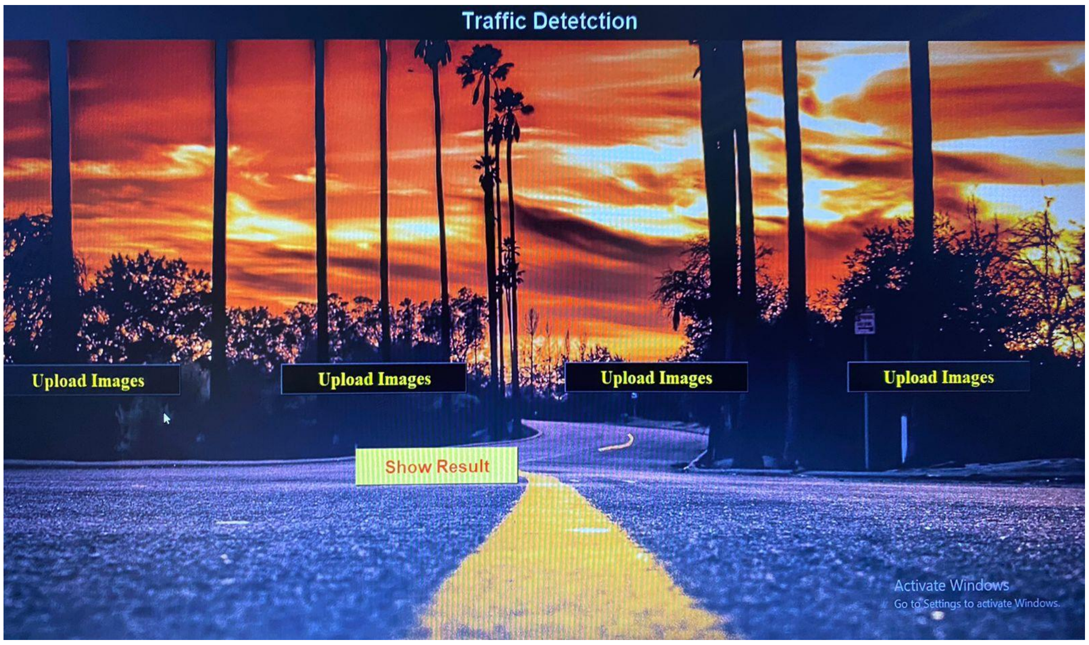


Figure: B.1 Traffic detection system-Front end Design

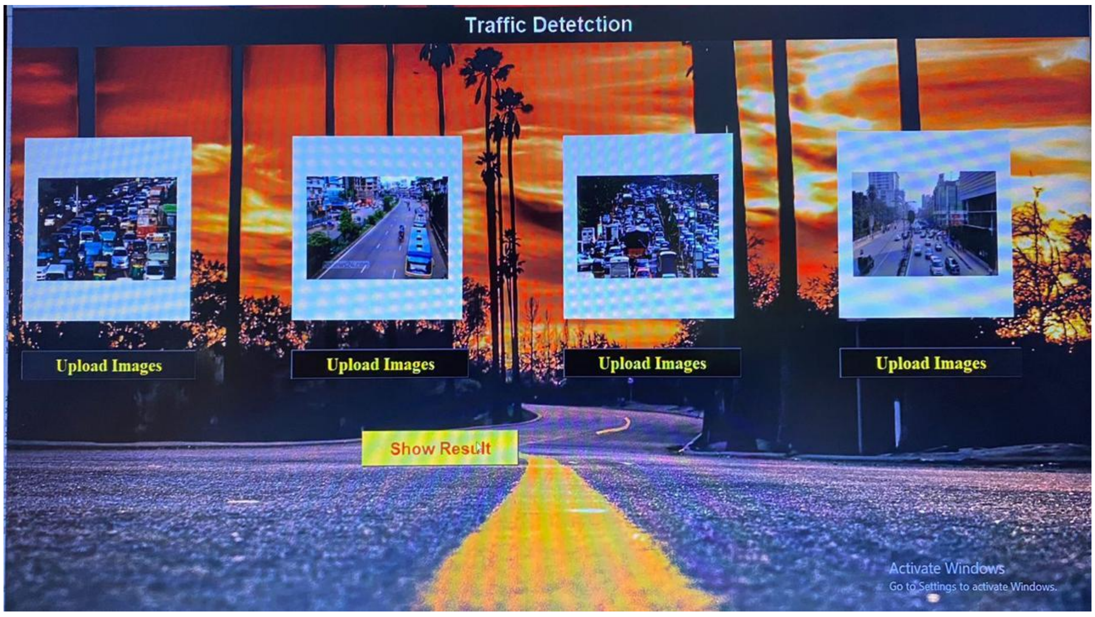


Figure: B.2 Image Uploading

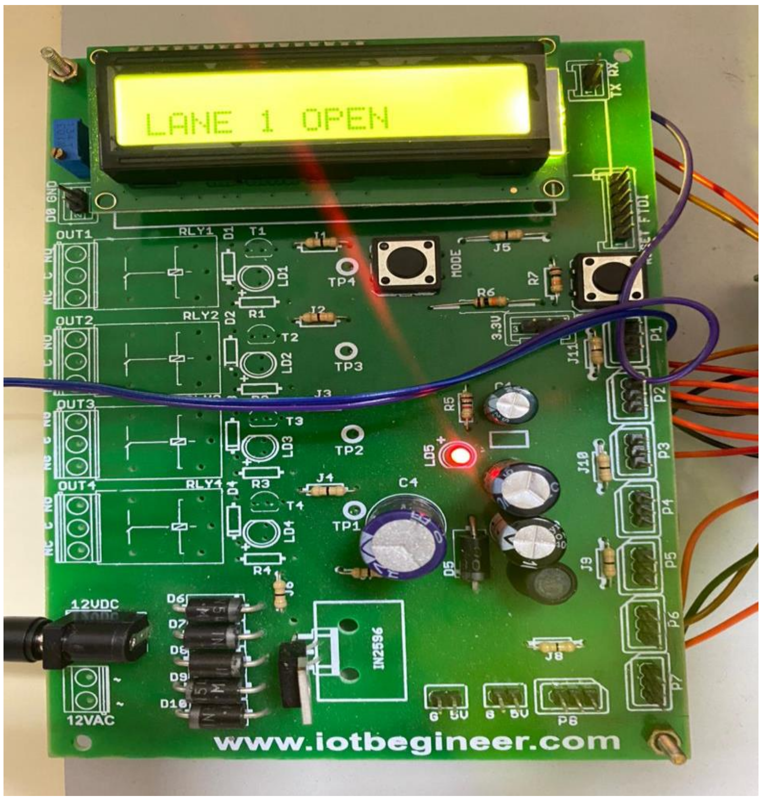


Figure: B.3 Integrated Board

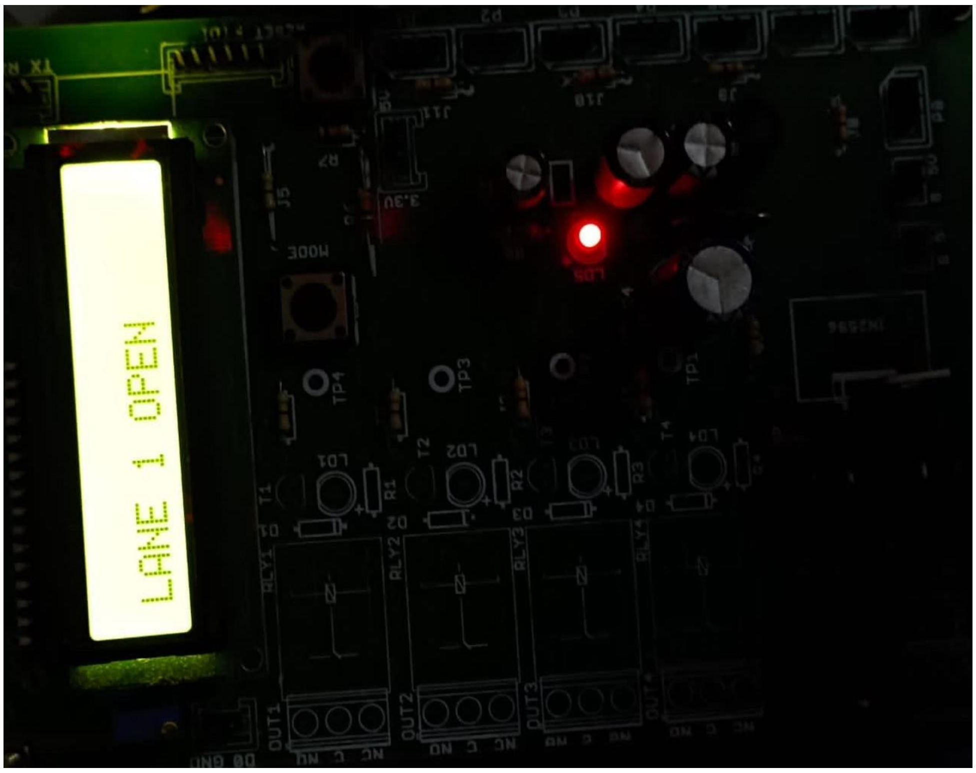


Figure: B.4 LANE 1 Open

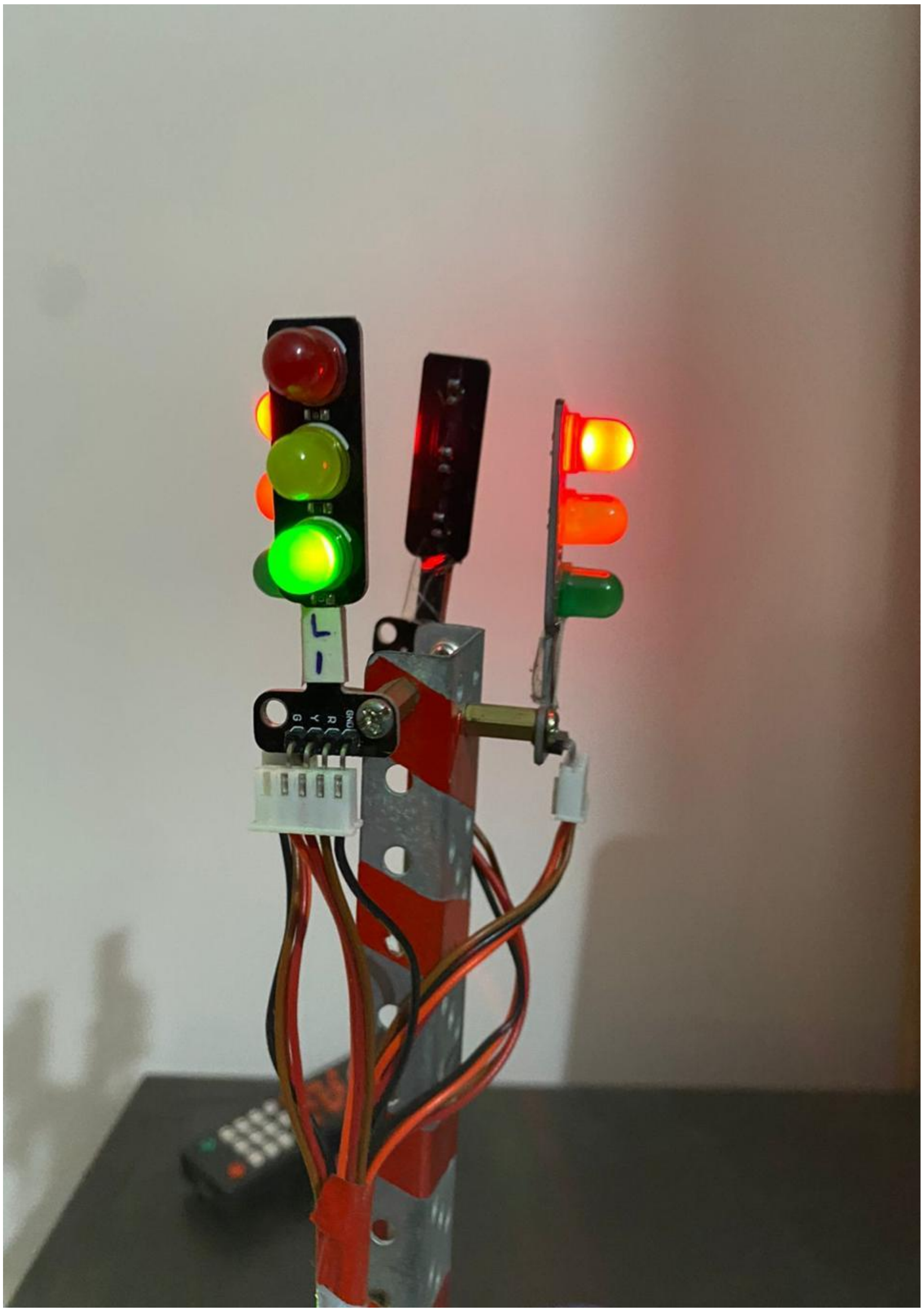


Figure: B.5 Traffic Light Set-up-Lane1

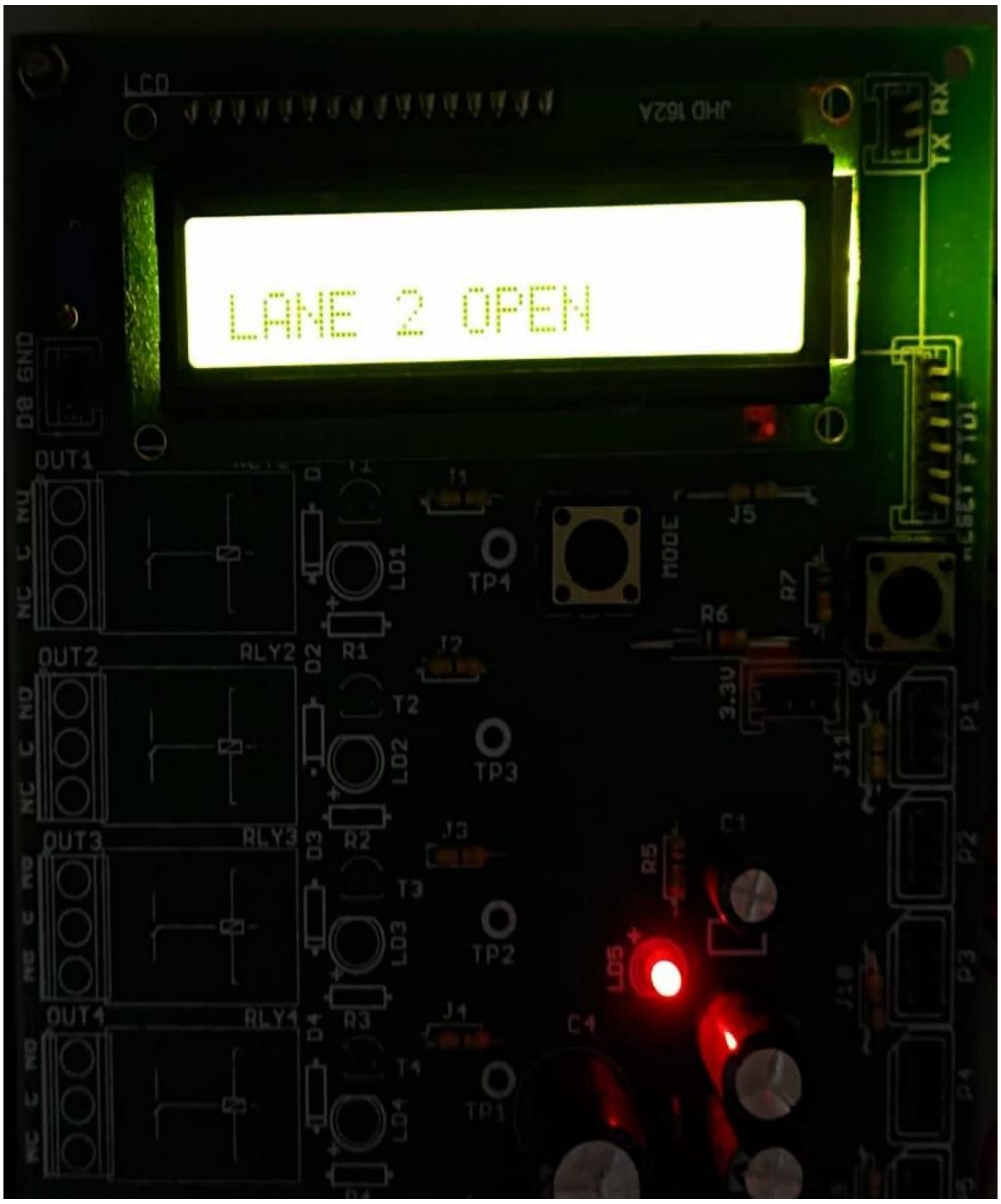


Figure: B.6 LANE 2 Open

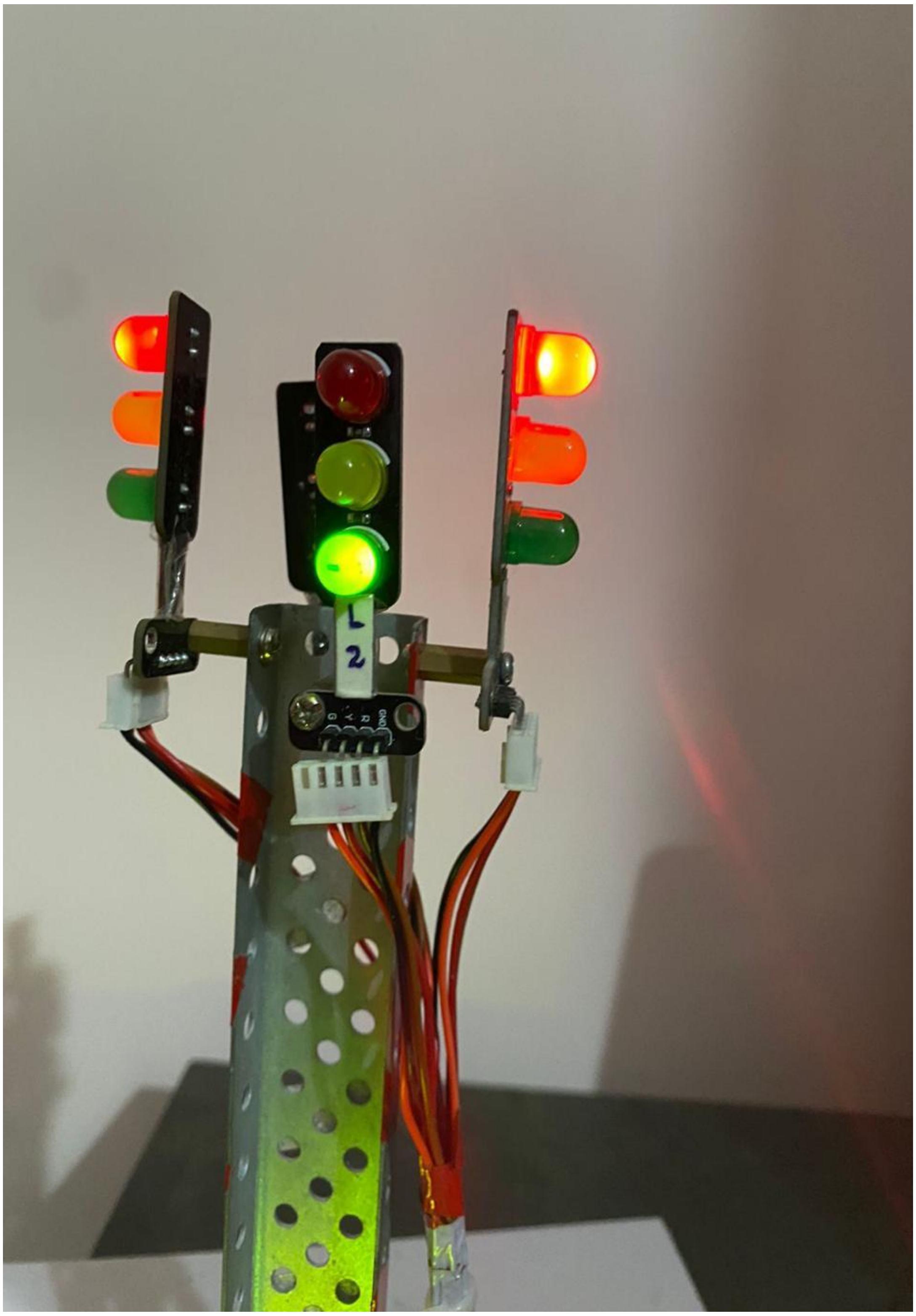


Figure: B.7 Traffic Light Set-up-Lane 2

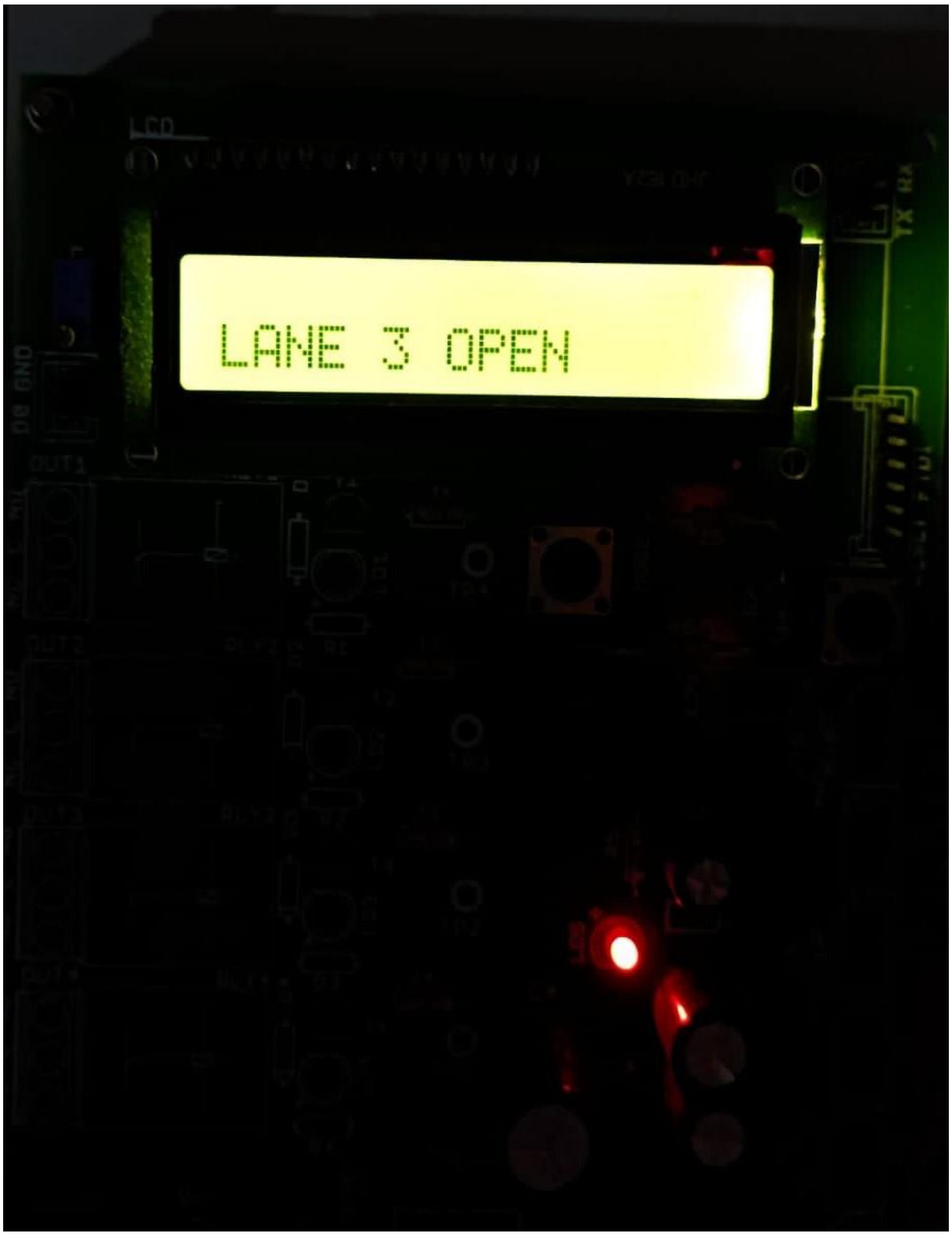


Figure: B.8 LANE 3 Open

Figure: B.9 Traffic Light Set-up-Lane 3

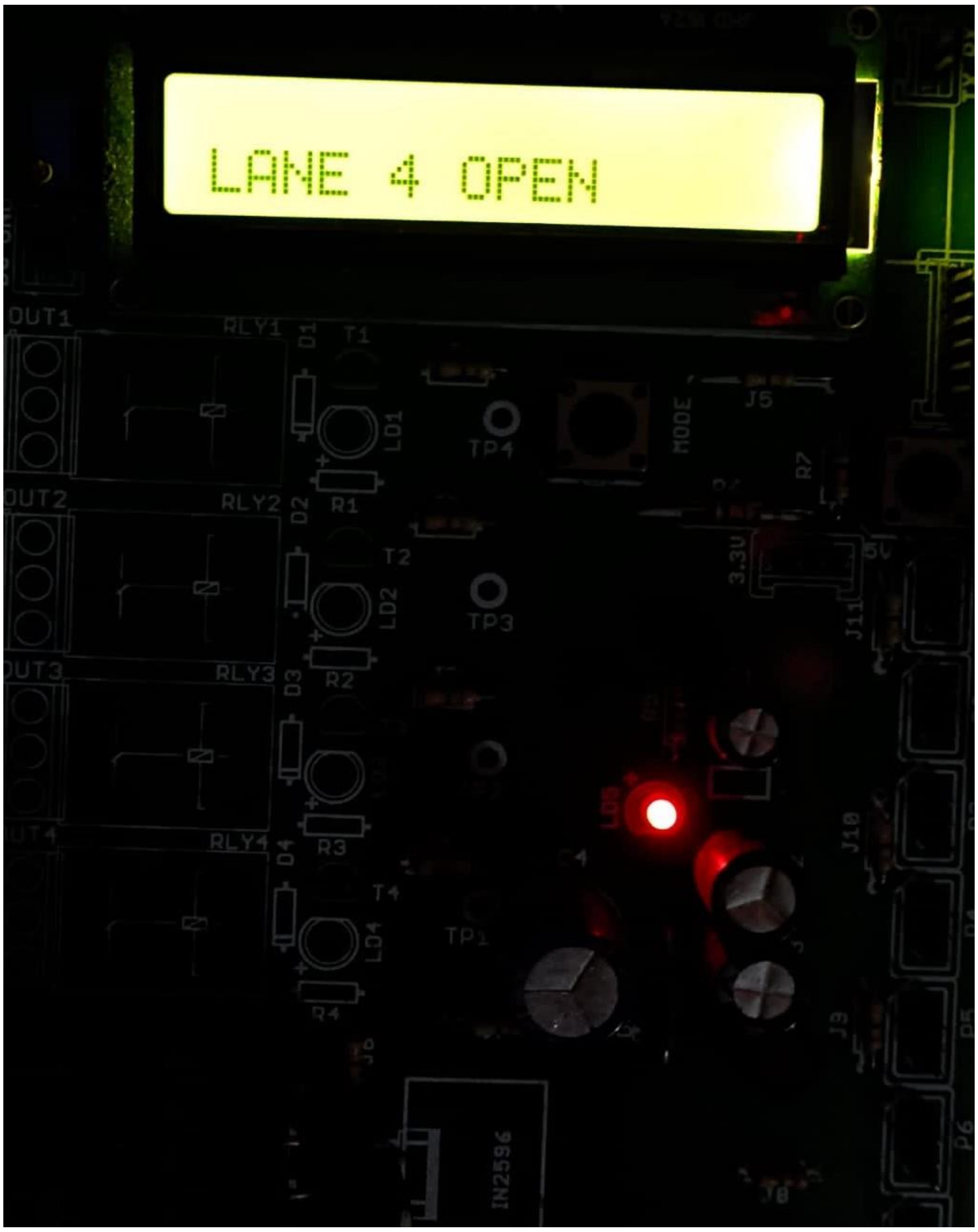


Figure: B.10 LANE 4 Open

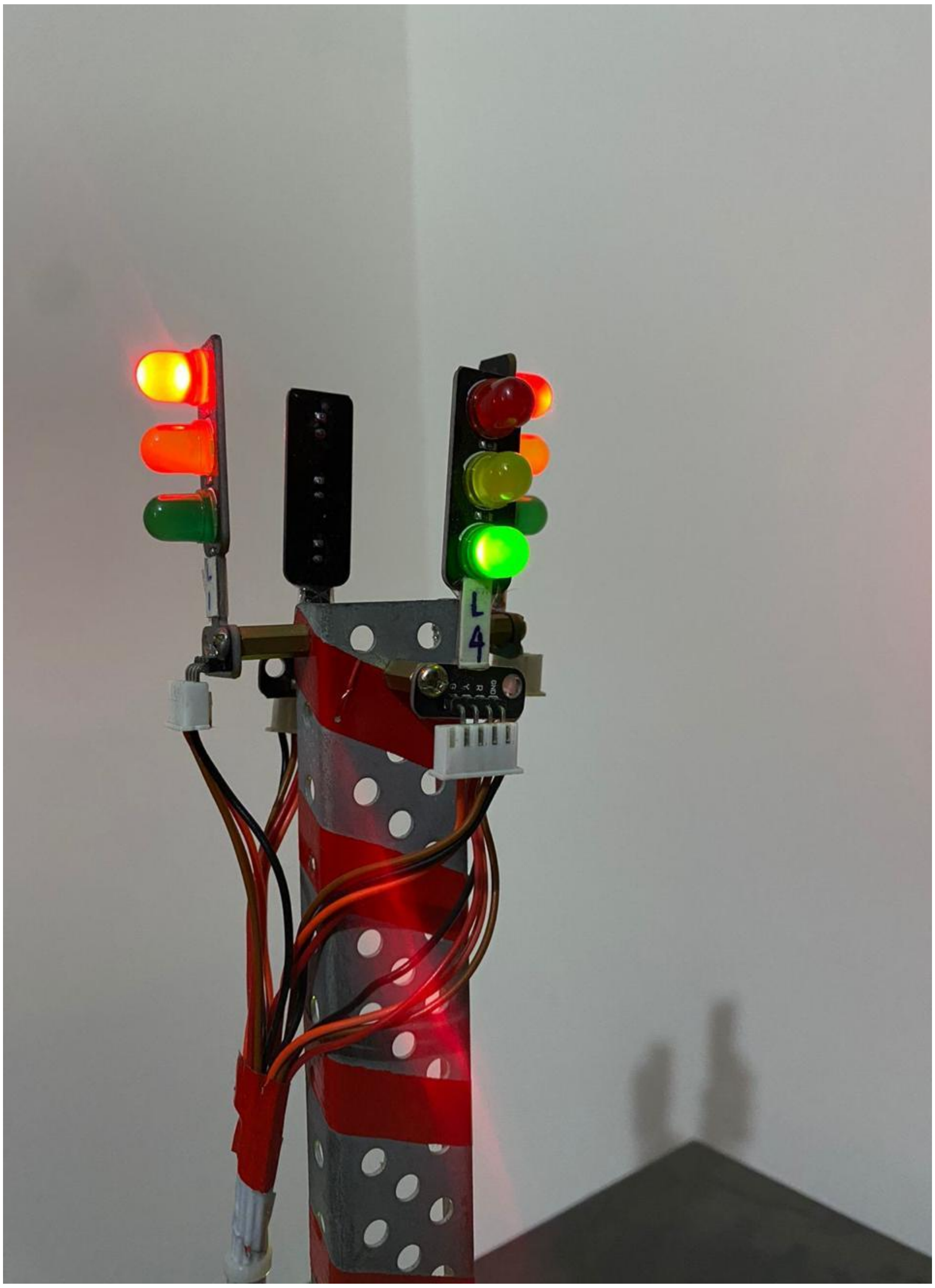
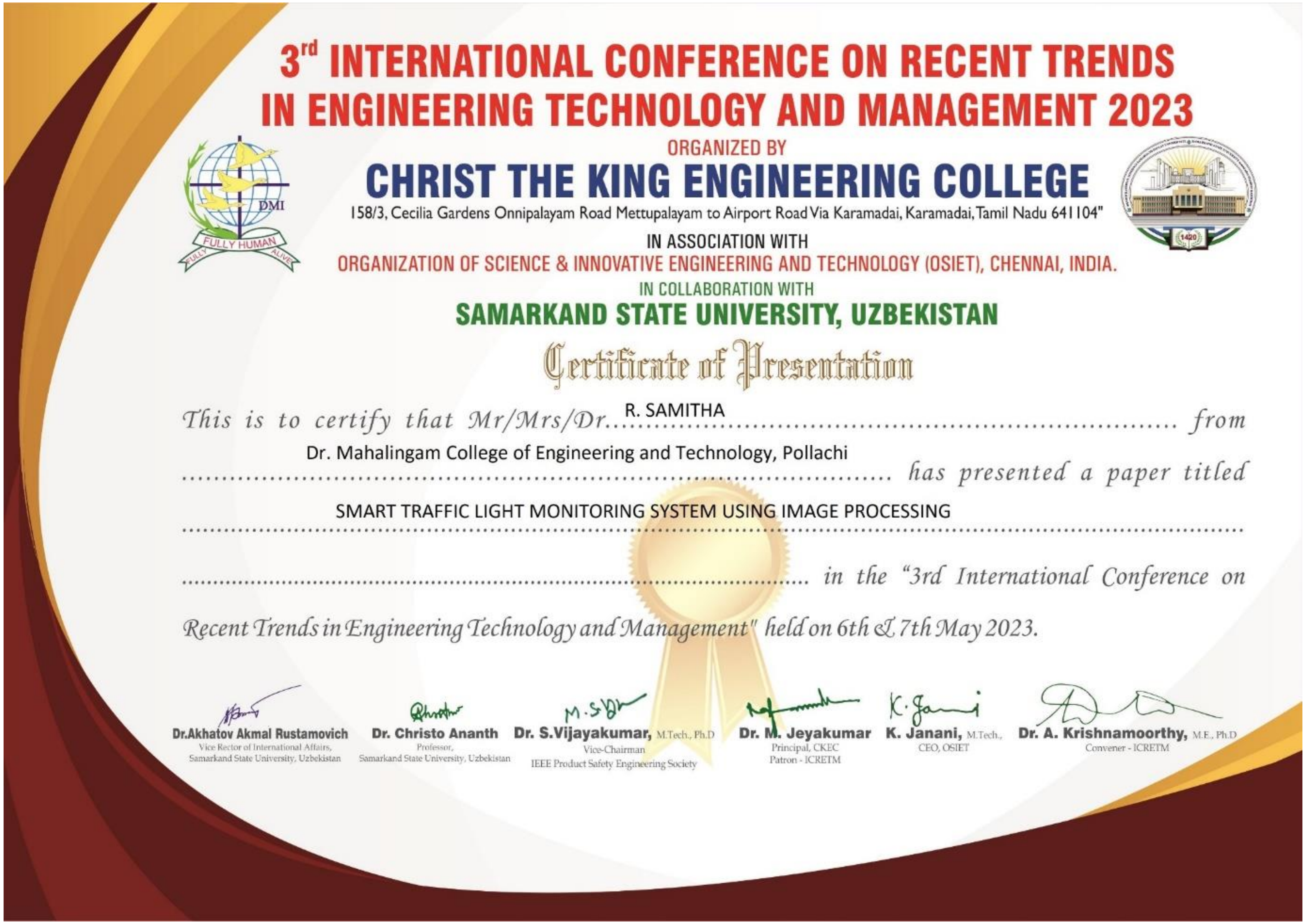


Figure: B.11Traffic Light Set-up-Lan



3rd INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ENGINEERING TECHNOLOGY AND MANAGEMENT 2023

ORGANIZED BY

CHRIST THE KING ENGINEERING COLLEGE

158/3, Cecilia Gardens Onnipalayam Road Mettupalayam to Airport Road Via Karamadai, Karamadai, Tamil Nadu 641104"

IN ASSOCIATION WITH

ORGANIZATION OF SCIENCE & INNOVATIVE ENGINEERING AND TECHNOLOGY (OSIET), CHENNAI, INDIA.

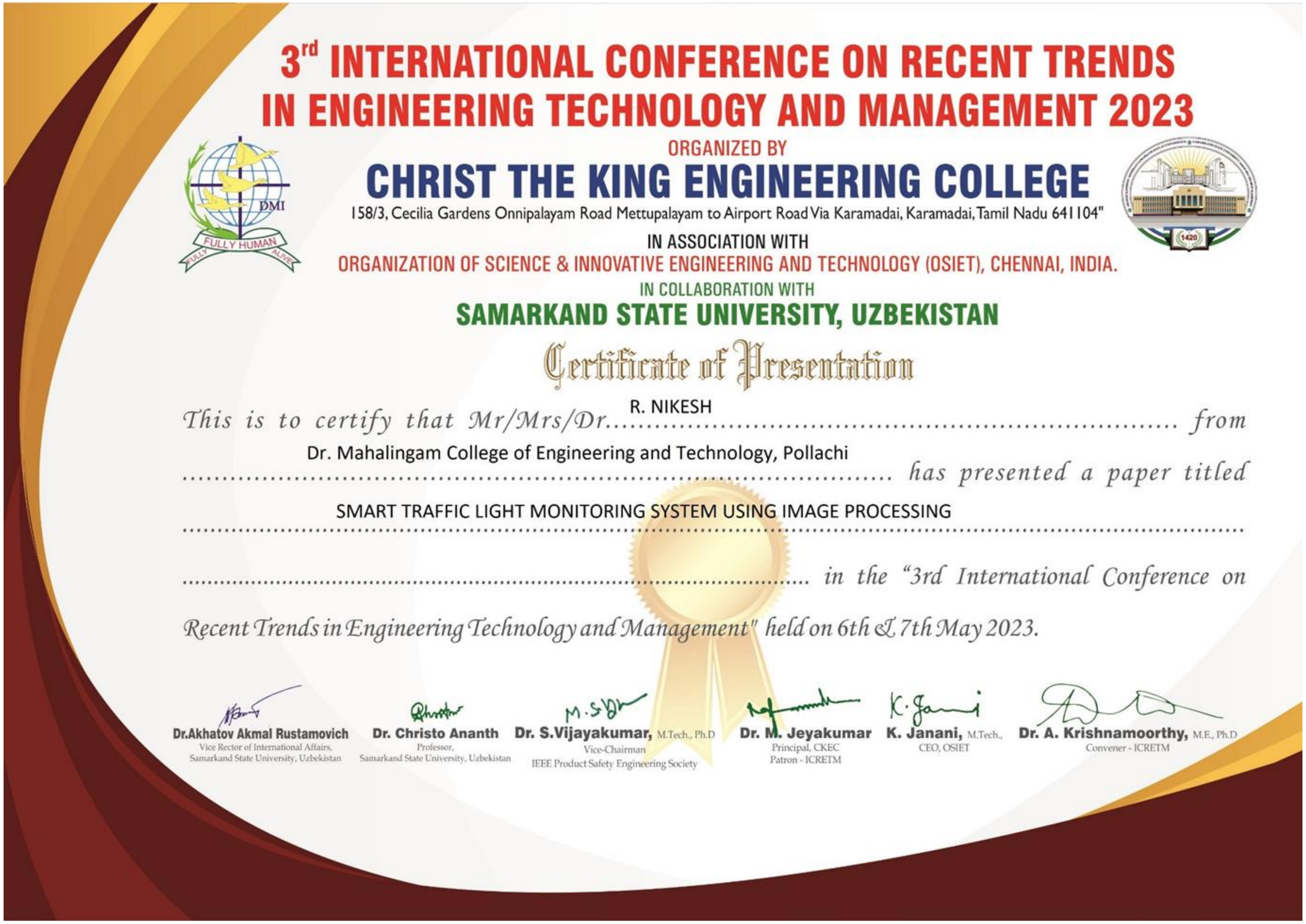
IN COLLABORATION WITH

SAMARKAND STATE UNIVERSITY, UZBEKISTAN

Certificate of Presentation

This is to certify that Mr/Mrs/Dr. R. SAMITHA *from* Dr. Mahalingam College of Engineering and Technology, Pollachi *has presented a paper titled* SMART TRAFFIC LIGHT MONITORING SYSTEM USING IMAGE PROCESSING *in the "3rd International Conference on Recent Trends in Engineering Technology and Management" held on 6th & 7th May 2023.*

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Dr.Akhatov Akmal Rustamovich	**Dr. Christo Ananth**	**Dr. S.Vijayakumar,** M.Tech., Ph.D	**Dr. M. Jeyakumar**	**K. Janani,** M.Tech.,	**Dr. A. Krishnamoorthy,** M.E., Ph.D
Vice Rector of International Affairs, Samarkand State University, Uzbekistan	Professor, Samarkand State University, Uzbekistan	Vice-Chairman IEEE Product Safety Engineering Society	Principal, CKEC Patron - ICRETM	CEO, OSIET	Convener - ICRETM



3rd INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ENGINEERING TECHNOLOGY AND MANAGEMENT 2023

ORGANIZED BY

CHRIST THE KING ENGINEERING COLLEGE

158/3, Cecilia Gardens Onnipalayam Road Mettupalayam to Airport Road Via Karamadai, Karamadai, Tamil Nadu 641104"

IN ASSOCIATION WITH

ORGANIZATION OF SCIENCE & INNOVATIVE ENGINEERING AND TECHNOLOGY (OSIET), CHENNAI, INDIA.

IN COLLABORATION WITH

SAMARKAND STATE UNIVERSITY, UZBEKISTAN

Certificate of Presentation

This is to certify that Mr/Mrs/Dr.......R. NIKESH...... *from* ......Dr. Mahalingam College of Engineering and Technology, Pollachi...... *has presented a paper titled* ......SMART TRAFFIC LIGHT MONITORING SYSTEM USING IMAGE PROCESSING...... *in the "3rd International Conference on Recent Trends in Engineering Technology and Management" held on 6th & 7th May 2023.*

[illegible] [illegible] M.S.[illegible] [illegible] K. Janani [illegible]

Dr.Akhatov Akmal Rustamovich
Vice Rector of International Affairs, Samarkand State University, Uzbekistan

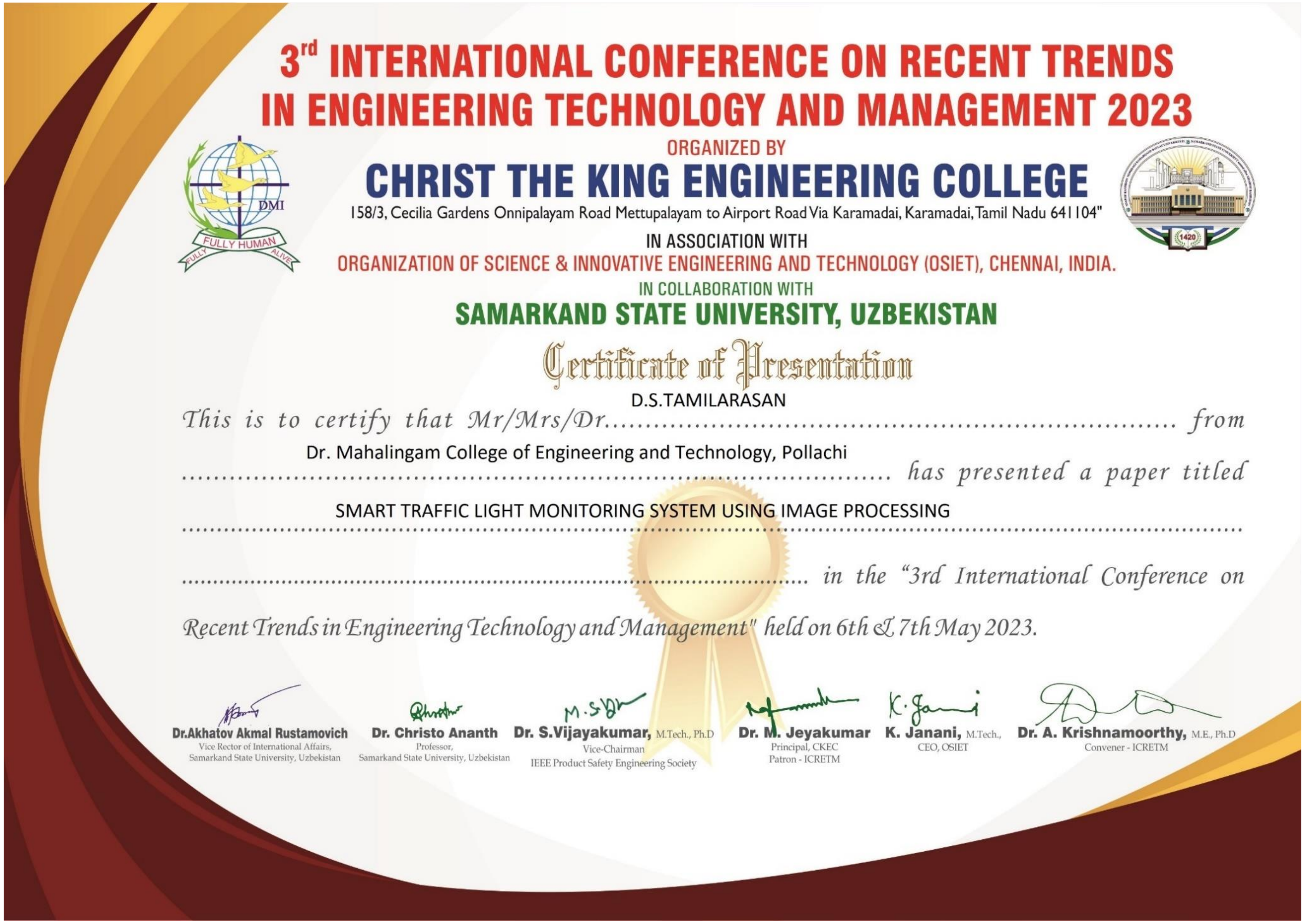
Dr. Christo Ananth
Professor, Samarkand State University, Uzbekistan

Dr. S.Vijayakumar, M.Tech., Ph.D
Vice-Chairman
IEEE Product Safety Engineering Society

Dr. M. Jeyakumar
Principal, CKEC
Patron - ICRETM

K. Janani, M.Tech.,
CEO, OSIET

Dr. A. Krishnamoorthy, M.E., Ph.D
Convener - ICRETM



3rd INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ENGINEERING TECHNOLOGY AND MANAGEMENT 2023

ORGANIZED BY

CHRIST THE KING ENGINEERING COLLEGE

158/3, Cecilia Gardens Onnipalayam Road Mettupalayam to Airport Road Via Karamadai, Karamadai, Tamil Nadu 641104"

IN ASSOCIATION WITH

ORGANIZATION OF SCIENCE & INNOVATIVE ENGINEERING AND TECHNOLOGY (OSIET), CHENNAI, INDIA.

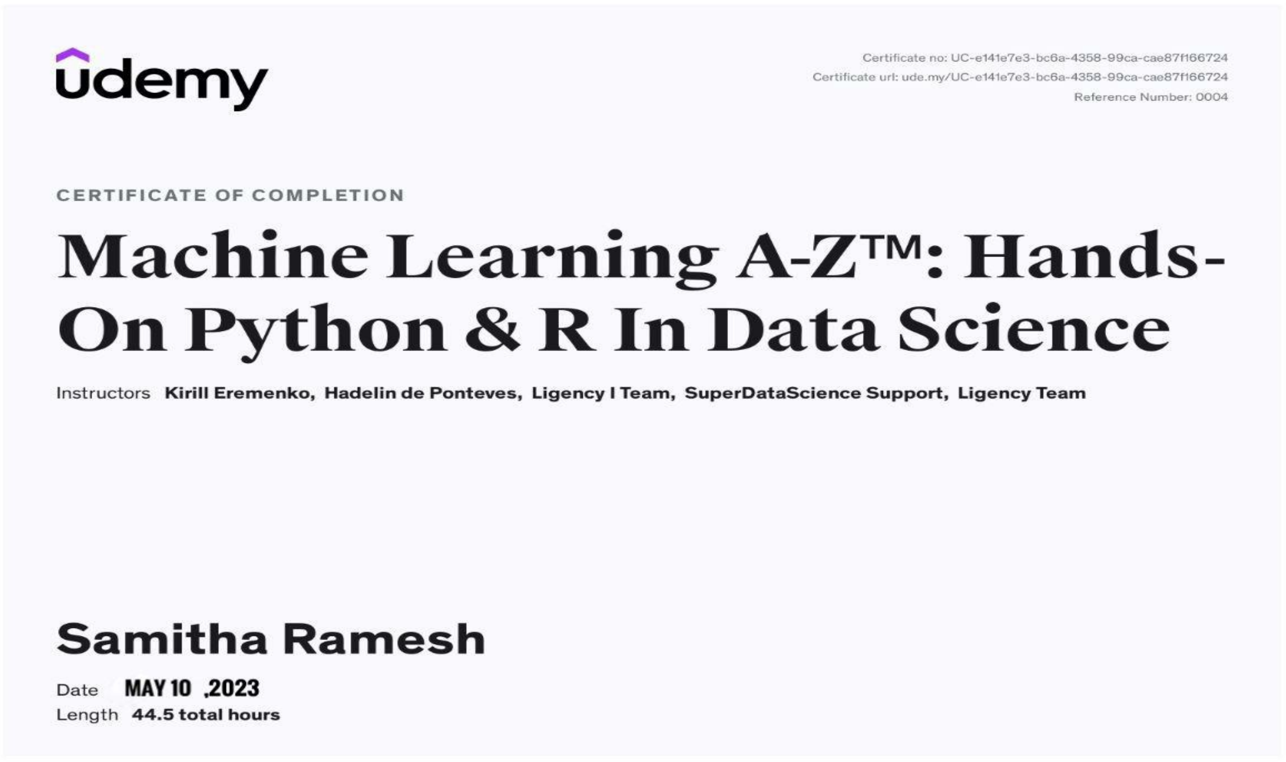
IN COLLABORATION WITH

SAMARKAND STATE UNIVERSITY, UZBEKISTAN

Certificate of Presentation

This is to certify that Mr/Mrs/Dr. D.S.TAMILARASAN *from* Dr. Mahalingam College of Engineering and Technology, Pollachi *has presented a paper titled* SMART TRAFFIC LIGHT MONITORING SYSTEM USING IMAGE PROCESSING *in the "3rd International Conference on Recent Trends in Engineering Technology and Management" held on 6th & 7th May 2023.*

Dr.Akhatov Akmal Rustamovich	**Dr. Christo Ananth**	**Dr. S.Vijayakumar,** M.Tech., Ph.D	**Dr. M. Jeyakumar**	**K. Janani,** M.Tech.,	**Dr. A. Krishnamoorthy,** M.E., Ph.D
Vice Rector of International Affairs, Samarkand State University, Uzbekistan	Professor, Samarkand State University, Uzbekistan	Vice-Chairman IEEE Product Safety Engineering Society	Principal, CKEC Patron - ICRETM	CEO, OSIET	Convener - ICRETM


udemy

Certificate no: UC-e141e7e3-bc6a-4358-99ca-cae87f166724
Certificate url: ude.my/UC-e141e7e3-bc6a-4358-99ca-cae87f166724
Reference Number: 0004

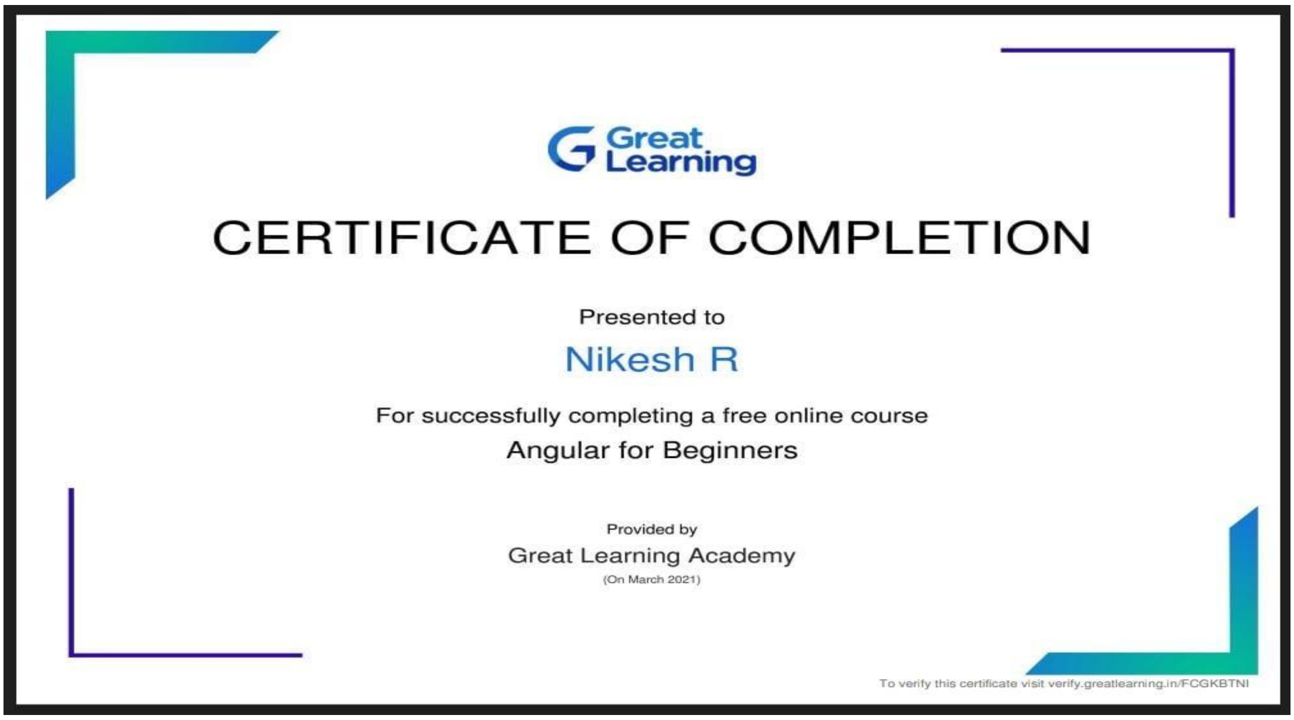
CERTIFICATE OF COMPLETION

Machine Learning A-Z™: Hands-On Python & R In Data Science

Instructors **Kirill Eremenko, Hadelin de Ponteves, Ligency I Team, SuperDataScience Support, Ligency Team**

Samitha Ramesh

Date **MAY 10 ,2023**
Length **44.5 total hours**


Great Learning

CERTIFICATE OF COMPLETION

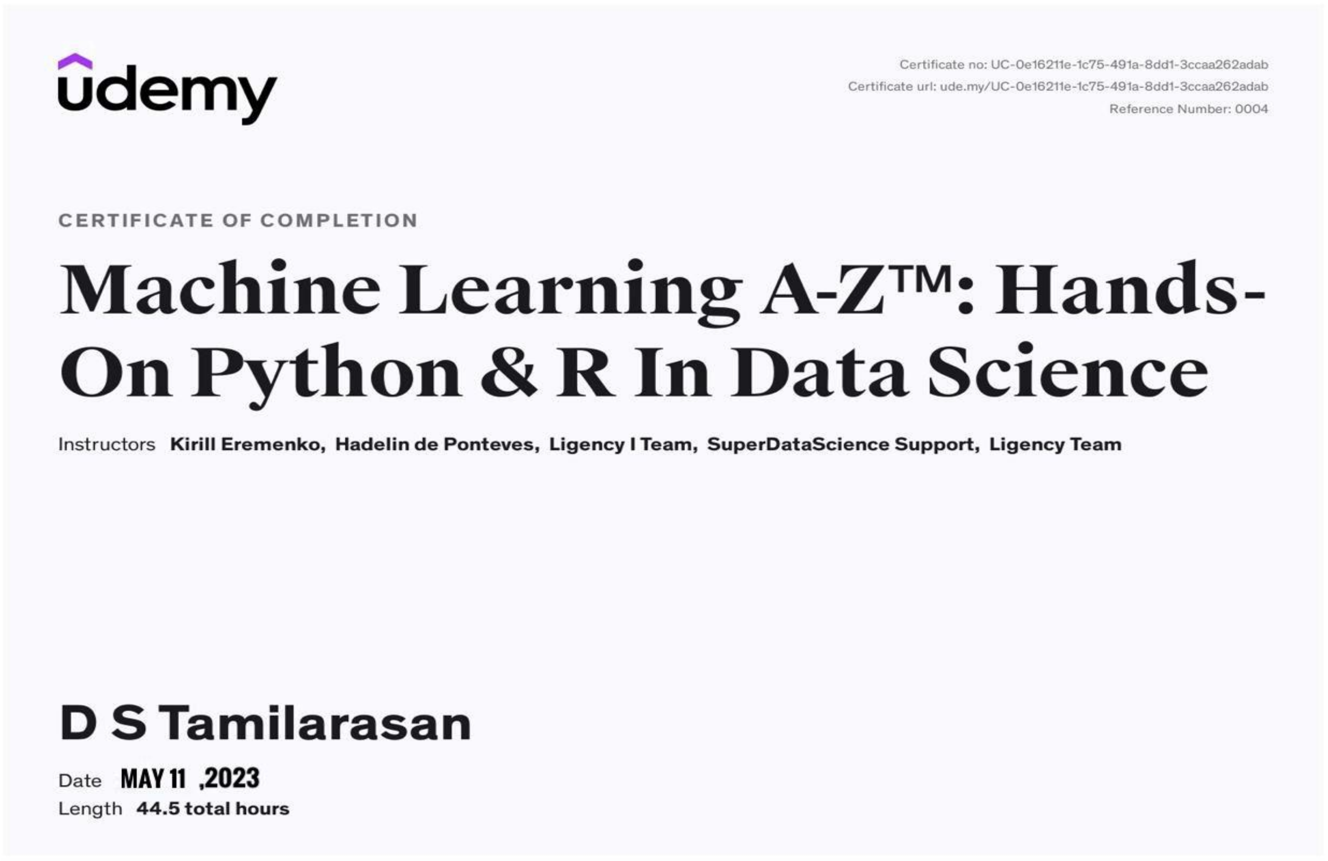
Presented to

Nikesh R

For successfully completing a free online course
Angular for Beginners

Provided by
Great Learning Academy
(On March 2021)

To verify this certificate visit verify.greatlearning.in/FCGKBTNI



udemy

Certificate no: UC-0e16211e-1c75-491a-8dd1-3ccaa262adab
Certificate url: ude.my/UC-0e16211e-1c75-491a-8dd1-3ccaa262adab
Reference Number: 0004

CERTIFICATE OF COMPLETION

Machine Learning A-Z™: Hands-On Python & R In Data Science

Instructors **Kirill Eremenko, Hadelin de Ponteves, Ligency I Team, SuperDataScience Support, Ligency Team**

D S Tamilarasan

Date **MAY 11 ,2023**
Length **44.5 total hours**